**1. Welcome and Personal Introduction (3–4 minutes)**

Good morning, everyone and thank you for being here today. My name is Mayank Saraf, and I am currently working on a very exciting internship project at Pyramid Cyber Security & Forensic (P) Ltd. that has been both challenging and rewarding.

Today, I will be sharing the journey of my project how I identified a real-world problem in the cybersecurity domain, how I designed a solution, and how I am building a web-based application to make security audits easier, faster, and more efficient.

I will walk you through the idea, the features, the technical side, and the potential impact this project can have. I will also share some behind-the-scenes stories and real-life examples that shaped my thinking.

Before we begin, I invite you to think about one question:   
**Have you or someone you know ever struggled with documentation-heavy, manual audit work?**   
If yes, you are going to relate to what I am about to share.

**2. The Problem in the Real World (6–7 minutes)**

Let us take a moment to think about the phrase (PAUSE) **"security audit."**   
What comes to mind for most people is a process full of documents, checklists, approvals, and, quite frankly, stress.

Now, I want to tell you a quick story that helped shape this project.

During my early research phase, I spoke with a cybersecurity analyst from a mid-sized IT company. They were preparing for an ISO 27001 internal audit. Their audit process involved five different Excel spreadsheets, a 90-page Word document listing controls, hundreds of email threads, and evidence files stored in various Google Drive folders. No central dashboard. No workflow. No automation.

The audit was due in four weeks. But three weeks in, they were still trying to match evidence to controls and figuring out who was responsible for what. It was not because they were inexperienced. It was simply because the process was too fragmented.

That story stuck with me because it showed that (STRESS HERE) the problem is not the people it is the process.

And this is not limited to one company. Across industries whether it is IT, healthcare, finance, or education the same challenges show up again and again.

Let me paint a picture.

Imagine(PAUSE) a security team preparing for an external audit. They have:

* A shared spreadsheet with hundreds of rows.
* A dozen people working on different sections.
* A shared drive filled with documents named “Final\_Final\_V3.”
* Emails like “Please upload the missing evidence ASAP.”
* A deadline in 5 days.

Now, imagine that the auditor finds mismatches, missing files, or duplicated entries. The team scrambles. Not because they did not do the work—but because they could not keep everything organized. The (STRESS HERE) process failed them.

This is how security audits become (STRESS HERE) a burden rather than a strategic opportunity to improve.

You might wonder why has this not been fixed already?

There are a few reasons:

1. **Over-reliance on generic tools:** Most organizations still use Excel, Word, and email. These are great tools but not designed for audits.
2. **Existing solutions are too expensive or too complex:** Enterprise audit software exists, but it often requires training, consulting, or large budgets making it inaccessible for smaller organizations.
3. **Fear of change:** Many teams are used to their current process. Even if it is inefficient, they continue with it because switching feels like more work.

But the consequences are serious:

* Missed compliance requirements
* Audit delays or failures
* Increased stress for teams
* Loss of trust from clients or regulators
* Security vulnerabilities going unnoticed

And ultimately, organizations spend more time (STRESS HERE) managing the audit than actually managing the risk.

Now I want to turn it over to you for a moment.

**Audience prompt:**   
Have you or someone you know ever experienced something like this?   
Where an audit process became chaotic simply because there was no central system or structure?

(STRESS HERE) This is the gap I saw.   
This is what motivated me to act.   
And this is what my application is built to solve.

**3. The Vision and the Idea (5–6 minutes)**

After identifying the real-world problems with current security audit processes the disorganization, the manual effort, the repetitive documentation I asked myself a simple question: (STRESS HERE) What would a better way look like?

At first, I just wanted to improve one small thing maybe help organize checklist items better. But the more I thought about it, the more I realized something deeper:   
(STRESS HERE) This is not a checklist problem. (STRESS HERE) This is a system problem.

The traditional approach to audits is reactive and document-heavy. People wait for the audit season, scramble to gather paperwork, and then breathe a sigh of relief when it is over. But security and compliance are not one-time events. They are ongoing responsibilities.

So I started thinking about what the experience should be instead.

Let me paint a new picture:

* Instead of Excel sheets scattered across email threads, you have a single dashboard where every audit task is tracked.
* Instead of remembering where the evidence was stored or who was responsible, everything is assigned, labelled, and time-stamped.
* Instead of spending a week building a report manually, it is generated with a single click organized, professional, and fully aligned to your selected compliance framework.
* Instead of being overwhelmed, teams feel in control.

That is when the vision came together.

I imagined a platform that brings (STRESS HERE) structure to chaos, simplifies audits from end to end, and turns a frustrating process into a manageable, even empowering, experience.

It would be built around three key ideas:

1. **Clarity** – The user always knows what to do, what is done, and what is next.
2. **Simplicity** – The tool does not add complexity; it removes it.
3. **Flexibility** – It adapts to different compliance frameworks and team workflows.

Here is a small analogy I like to use:

Think of traditional audits like building a house using only handwritten instructions, sticky notes, and phone calls.   
Now imagine doing it with a proper project management platform, a blueprint, shared task boards, and real-time status updates.   
Which one would you prefer?

This was not just about building software. It was about (STRESS HERE) reimagining how people approach security and compliance.

I envisioned something that even a junior team member could use without fear. Something that a compliance lead could trust. Something that works **with people**, not against them.

That became the foundation of my project:   
A web-based application that brings together (STRESS HERE) audit management, evidence tracking, collaboration, and reporting all in one place.

This vision came not just from observing problems but from imagining the relief and confidence that users would feel when using a better system.

**4. Application Features with Real-Life Scenarios (10–12 minutes)**

Now let me take you on a guided walkthrough of what my application actually does.   
Instead of showing only the technical features, I want to show you how each feature helps (STRESS HERE) real people solve (STRESS HERE) real problems.

Imagine you are part of a cybersecurity team at a fast-growing company. You have limited time, multiple audits to prepare for, and no centralized system. Now, here is how my application makes your job easier:

**1. Dynamic Audit Checklist Builder**

Traditional method: Teams copy controls from documents, paste them into spreadsheets, and edit them over and over for each audit cycle.   
Result? Inconsistency. Rework. Confusion.

**My approach:**   
My application allows you to (STRESS HERE) create and reuse custom checklist templates for different standards like ISO, NIST, or PCI-DSS. You can select controls, add descriptions, mark priorities, and even link recommended evidence formats.

**Real-life scenario:**   
A compliance officer preparing for an ISO 27001 internal audit builds the checklist in minutes using predefined controls. For next year’s audit, they just clone and update it saving days of repetitive work.

**Impact:**

* No more copy-paste errors
* Standardization across audits
* Faster preparation

**2. Task Assignment and Role Management**

In typical audits, the question “Who is responsible for this?” comes up again and again.   
People forget, things fall through the cracks, and accountability suffers.

**My approach:**   
Each checklist item can be (STRESS HERE) assigned to a team member, with a due date and status tracker. You can also define user roles admin, auditor, viewer to control visibility.

**Scenario:**   
Let’s say your IT manager is responsible for network security policies, while HR handles employee onboarding checks. Both receive their tasks directly, get reminders, and update the status from their dashboard. You, as the audit lead, can see everyone’s progress in one place.

**Impact:**

* Clear ownership
* No missed tasks
* Better teamwork

**Audience prompt:**   
Have you ever had to chase people over email for audit responses?   
Imagine a tool that removes that need entirely.

**3. Secure Evidence Upload and Storage**

Auditors need to gather screenshots, policy documents, logs, and certifications as evidence. These are usually shared through drives, emails, or shared folders.

**Problem:**   
Disorganized. Insecure. Hard to trace back later.

**My approach:**   
Each checklist item in my application has a built-in (STRESS HERE) secure evidence upload panel. Files are encrypted at rest, and access is restricted based on roles. You can preview, download, or replace evidence without leaving the page.

**Scenario:**   
A data privacy control requires proof of access logs. Your IT team uploads a CSV log directly to that control. The auditor sees it, marks it reviewed, and moves on all within the app.

**Impact:**

* One secure place for all evidence
* No need for back-and-forth emails
* Easy audit trail for future reference

**4. Live Compliance Dashboard**

When teams use spreadsheets, understanding audit progress is difficult. You have to manually count completed rows or apply filters.

**My approach:**   
The dashboard in my application shows (STRESS HERE) real-time compliance metrics.

* Percentage complete
* Number of pending tasks
* Framework-specific heatmaps
* Controls failing or overdue

**Scenario:**   
During a status meeting, the audit lead shares their screen and opens the dashboard. In seconds, the team sees what is done, what is delayed, and what needs urgent attention. No manual report building required.

**Impact:**

* Instant insights
* Better planning
* Confidence in progress

**5. One-Click Report Generator**

End-of-audit reporting is one of the most time consuming steps. People copy data from checklists, format tables, convert PDFs and write summaries.

**My approach:**   
At the end of the audit, my application generates a structured PDF report with:

* Compliance summary
* Task completion status
* Linked evidence
* Notes from reviewers
* Timestamp and framework labels

**Scenario:**   
A consultant completes a PCI-DSS audit for a client. Instead of spending a full day creating the report, they click “Generate Report” and deliver it in minutes.

**Impact:**

* Saves time
* Professional, consistent output
* Reduces post-audit chaos

**6. Interviewer Mode – Guided Q&A Audits *(Flagship feature)***

Here is something that truly sets my application apart.

**Problem:**   
New teams, interns, or small companies often feel lost during audits. Reading long checklists is mentally tiring. Mistakes happen.

**My approach:**   
“Interviewer Mode” turns the audit process into a (STRESS HERE) guided Q&A. Think of it as a digital assistant asking one question at a time, explaining context, and prompting you for evidence.

**Scenario:**   
A junior intern is assigned to conduct a first-pass audit. Instead of being overwhelmed, they follow the conversational flow:

* Question: “Do you have a password policy in place?”
* Tip: “This can be a PDF of your internal IT policy.”
* Upload: *Drag file here*

(STRESS HERE) It feels like a conversation, not a chore.

**Impact:**

* Reduces cognitive load
* Makes audits beginner-friendly
* Speeds up onboarding of new team members

Each of these features was designed to solve a (STRESS HERE) specific, real-world problem I saw during research and interviews.   
They are not just software elements they are responses to pain points felt by teams everywhere.

From checklist creation to evidence collection, from team collaboration to final reporting my application transforms the entire audit experience.

It is more than a tool.   
It is a shift in how we approach compliance.

**5. Technical Walkthrough – Behind the Scenes of My Application (6–7 minutes)**

Now that I have shown you what the application *does*, let me take you behind the scenes and explain (STRESS HERE) how I built it and more importantly, (STRESS HERE) why I made the decisions I did.

This section will give you a look at the technology stack, architecture, and some key development choices that shaped the way the application works.

**The Tech Stack: Modern, Scalable, and Enterprise-Ready**

Alright, now let’s talk about the technological backbone of the cybersecurity audit web application we’ve built. This is one of my favorite parts to discuss, so stay with me, and feel free to reflect on how some of these decisions might apply to your own projects or future enterprise systems.

Let me start with a question: Have you ever worked on a project where you had to juggle between outdated tools and limited scalability? It’s frustrating, right? Well, I made sure to avoid that. For this project, I was very intentional about choosing technologies that are modern, scalable, and well-supported in the enterprise landscape — and also accessible for small security teams or startups trying to implement structured cybersecurity audits without needing massive infrastructure investments.

Here is the core tech stack I used:

| **Component** | **Choice** |
| --- | --- |
| Frontend | Razor Pages & ASP.NET Core MVC |
| Backend | ASP.NET Core Web App (C#) |
| Database | PostgreSQL |
| IDE | Visual Studio / Visual Studio Code |
| Hosting | Azure (Free Tier) + Cloudflare CDN + Render/Railway (optional) |
| Programming Language | C# (primary), Python (auxiliary tooling only) |

Let’s break it down. And I’ll try to keep it interactive — so as I go through each part, think about what you would have chosen, and how your stack might differ.

**Frontend – Razor Pages with ASP.NET Core MVC**

Let’s begin with the frontend.

Now, here’s a question for all the developers or security architects in the room: how many of you have worked with single-page applications like React or Angular? They’re great, right? But sometimes, especially in compliance or audit applications, we need something more robust, more structured, and easier to maintain — with built-in support for routing, security, and server-side rendering.

That’s why I went with **Razor Pages** alongside **ASP.NET Core MVC**.

ASP.NET’s Razor Pages framework offers a **page-centric development model** — this is perfect for building enterprise-grade interfaces like audit dashboards, compliance workflows, form submissions, and admin control panels. The MVC component brings the power of separating logic, views, and controllers, which helps make the application easy to test, scale, and extend.

With Razor Pages, each page handles its logic and view independently. This is especially important in audit applications where each page — for example, a checklist view or evidence uploader — carries its own logic, permissions, and data bindings.

**Key reasons for choosing Razor Pages:**

* Full .NET integration with backend logic and security.
* Supports role-based rendering and access restrictions natively.
* Ideal for enterprise audit forms with nested data (e.g., control objectives inside frameworks).
* Fast rendering time and better SEO if needed for client portals.

Plus, with minimal frontend JS needed, it becomes easier to audit from a security perspective — fewer third-party JS dependencies mean fewer risks.

**Backend – ASP.NET Core Web App (C#)**

Moving on to the backend — this is where the real logic lives.

I used **ASP.NET Core Web Application in C#**. Now, why C#? That’s a great question. C# is powerful, strongly typed, and incredibly secure when it comes to handling sensitive backend logic. In an audit system where we deal with evidence files, reports, permissions, and regulatory frameworks, this level of reliability is essential.

The backend is responsible for:

* Serving and validating audit questions
* Managing users, roles, permissions
* Handling file uploads and secure evidence storage
* Generating compliance reports
* Mapping GRC (Governance, Risk, and Compliance) hierarchies
* Logging every action for audit trails

I implemented the entire backend using RESTful APIs, organized by entity: frameworks, domains, control objectives, etc. These endpoints communicate seamlessly with the frontend Razor Pages using model binding and form handlers.

Some other technical aspects:

* **Dependency Injection** is used throughout for better testability and cleaner logic.
* **Entity Framework Core** handles ORM (Object-Relational Mapping) and connects to our PostgreSQL database.
* **Middleware** handles JWT authentication, access control, and request logging.
* **Logging** is powered by Serilog, and error handling includes both developer-friendly and user-friendly messages.

And if you’re wondering — yes, everything is asynchronous and supports scalability.

**Database – PostgreSQL**

Ah, the database — the heart of any serious application.

For this project, I selected **PostgreSQL**. It’s an open-source relational database that is rock-solid, scalable, and trusted across industries. It offers robust performance, powerful SQL features, and enterprise-grade security.

Now, why PostgreSQL instead of a NoSQL database like MongoDB? Simple:

* Our data is **highly relational**. Audit questions belong to control objectives. Controls belong to security domains. Domains belong to frameworks. And so on.
* We needed strong data integrity with foreign keys and relational joins.
* PostgreSQL supports advanced queries and full-text search — useful for large frameworks like NIST or ISO.

I’ve modeled a full **GRC-style hierarchy** in the database:

1. **Domains** (e.g., Cybersecurity, Privacy, Risk)
2. **Frameworks** (e.g., NIST, ISO 27001, OWASP, PCI-DSS)
3. **Security Domains** (e.g., Access Control, Network Security)
4. **Controls**
5. **Control Objectives**
6. **Audit Questions**
7. **Evidence Items**
8. **Technical Implementations**

Each layer has foreign key relationships, and everything is normalized — meaning minimal duplication, fast queries, and easy reporting.

**Authentication and Role-Based Authorization**

Now, let’s talk about **security** — a core aspect of this project.

I implemented **JWT-based authentication**. When a user logs in, a token is generated and stored securely. Every request carries this token to validate the session. Roles and permissions are enforced using ASP.NET Core’s **Policy-Based Authorization**.

There are four main roles:

* **Admin** – full control of users, roles, and audit templates
* **Auditor** – can conduct audits, upload evidence, fill forms
* **Reviewer** – can verify audit entries and mark them as complete
* **Viewer** – can only read results and download reports

Access to each page and endpoint is guarded based on this role. Logs are maintained for every change made — we’re talking full **audit trail capability**, right from the backend.

**File Storage and Evidence Upload**

Now, how many of you have used or managed file upload systems before?

It’s not just about uploading — it’s about **security, traceability, and compliance**.

This application includes a full-fledged **evidence upload system**, where users can upload files related to each audit question.

Key features:

* Files are uploaded via secure endpoints (POST with anti-forgery tokens).
* Files are stored either in **Azure Blob Storage**, **Railway’s volume storage**, or securely on the same server (based on deployment).
* Files are encrypted **in transit** (HTTPS) and **at rest** (AES-256).
* Only users assigned to an audit can see or download the files.
* All file activity is logged with timestamp, user ID, file metadata.

There’s even a **basic malware scanner** (Python-based) that checks uploaded files for suspicious content, and **file-type restrictions** to prevent .exe or .bat uploads.

**Deployment and Hosting**

Let me ask you this — how many of you have tried deploying a .NET Core app to Azure? It’s surprisingly smooth!

I hosted the application using **Azure App Services** — it integrates perfectly with ASP.NET and offers free-tier hosting for testing and demos.

For DNS and performance, I use **Cloudflare** as a **CDN (Content Delivery Network)** and DNS provider. This helps improve response time and protect against DDoS attacks.

Alternative deploy options include **Railway** or **Render**, which can be used to host PostgreSQL and backend services with built-in CI/CD pipelines.

CI/CD is managed using **GitHub Actions**, which means:

* Every push to the main branch triggers a build and deployment.
* Any test failures stop the deploy.
* Logs are maintained for traceability.

This makes development smoother, especially with team collaboration.

**Supporting Emerging Technologies**

Now here’s the exciting part — this platform isn’t just a static checklist app.

It is designed to support **cybersecurity audits and compliance mapping** for both traditional and **emerging domains**, including:

* **Core Cybersecurity Domains**
* **Cloud Computing, Edge, and Fog Infrastructure**
* **Quantum Computing Risk & Controls**
* **IoT & Smart Devices Security**
* **Game Development Environments**
* **Augmented, Virtual, and Extended Reality (AR/VR/XR)**
* **Autonomous & Vehicular Technology**
* **Blockchain & FinTech Compliance**
* **Cryptographic Modules (FIPS, etc.)**
* **Bioinformatics & Genomic Data Protection**
* **Artificial Intelligence & Machine Learning**
* **Generative AI (e.g., Prompt Injection Threats)**
* **Big Data Analytics**

The idea is to have **customized control templates** that can be mapped for specialized sectors, ensuring the tool is future-proof and extensible.

**Performance Optimization and Monitoring**

To ensure fast load times, I added:

* Lazy loading of dashboard widgets
* Pagination for long checklists
* Cache management for user sessions and static content
* Error logging using console wrappers and alerts (basic implementation)

In the next version, I plan to integrate tools like Sentry or LogRocket for advanced monitoring.

The technology choices were not random. Every component was selected to match the mission of the platform:

* Easy to build and scale
* Secure and privacy-focused
* Simple enough for fast development
* Flexible enough for future upgrades

This architecture forms the foundation for all the features I showed you earlier — from checklist building to evidence upload to automated report generation.

What makes me most excited is that the structure is modular. If tomorrow I want to plug in AI, mobile apps, or external compliance APIs, the system can support it.

**6. Market Comparison – Why My Application is Different (2 minutes)**

Now let us look at how my solution compares with existing ones like:

* CyberMetric
* Drata
* RSA

While these platforms are powerful, they are:

* Expensive (pricing starts at thousands of dollars)
* Not easy to use without training
* Often locked into specific frameworks

My application is designed for:

* Simplicity
* Affordability
* Framework flexibility
* Easy onboarding within minutes

It is ideal for small and medium businesses, educational institutions, and cybersecurity consultants.

**7. Development Challenges and Learning Journey (2 minutes)**

Some key challenges I faced:

* Understanding different compliance standards
* Designing the database schema to support flexibility
* Securing file uploads against threats like malware
* Balancing simplicity and completeness

I overcame them by:

* Studying real audit reports from CyberMetric a complete compliance management product
* Reading OWASP documentation
* Getting feedback from mentors and users
* Iterating the UI multiple times

**8. Future Vision (2 minutes)**

This project is just the beginning. In future versions, I plan to add:

* AI to recommend missing tasks
* Alerts for compliance risks
* API integrations with Slack, Jira, and Google Drive
* Mobile-first design
* Audit timeline and export logs

The ultimate goal is to build a secure, intelligent audit platform that supports continuous compliance and security awareness.

**9. Final Thoughts and Closing (2 minutes)**

To close this session, I want to say this:

My application is not just a tool it is a response to a problem that exists everywhere: complexity in important tasks.

We live in a world where simplicity wins, and where technology should serve the user does

 not confuse them.

I built this application with the belief that anyone, regardless of technical expertise, should be able to carry out a high-quality security audit with confidence.

I hope my journey, my solution, and this session have inspired some new ideas and possibilities for your own work.

(STRESS HERE) Thank you very much for listening. I welcome your questions, suggestions, and feedback.