Project Timeline – APK Analyser Development

Each session was conducted every other week and shows how the project progressed from the beginning stages of planning to a full interactive web application.

Meeting 1 – Initiating the Project and First Steps

Focus: Established specific objectives and discovered mobile application security vulnerabilities with the OWASP Mobile Top 10.

Discussion: Originally suggested Ghidra for analysis but shifted towards manifest based APK inspection after considering practicality.

Outcome: We came up with a tool that scans for risky components, including exported activities and permissions.

Meeting 2 – Requirements Definition and Technical Research

Focus: Learned Android app structure and manifest format in detail.

Discussion: Investigated component types (activities, services, providers) and whether they are exported. Consulted Androguard documentation, Android Developers documentation, and static analysis labs.

Outcome: examine permissions and exported items through AndroidManifest.xml. Define deliverables to generate CSV exports of these results.

Meeting 3 – Choosing Tools and First Script

Focus: Implementation design and tool comparison.

Discussion: Chose Python with JADX to decompile APKs and parse manifests. I considered VirusTotal and MobSF but did not think they provided component level details.

Outcome: Created the first working command-line script to fetch and label exported activities/services and store them as raw CSV files.

Meeting 4 – Classification, Filtering, and Metadata

Focus: Enhanced data extraction quality.

Discussion: Started filtering with Pandas. Began marking components with their export status. Added fields like API level and APK signature.

Outcome: Generated formatted CSV reports. Started to differentiate dangerous vs normal permissions manually.

Meeting 5 – Internal Database Preparation and Validation

Focus: Depth of classification and automation potential.

Discussion: Created a static dictionary based on research (such as Stack Overflow) to label permission levels. Researched the possibility of obtaining live updates by scraping Android's website.

Outcome: CSVs now include API level, app signature, permission name, type, services and providers in separate filtered CVSs. Started creating automation plans.

Meeting 6 – Comparative Analysis and Tool Uniqueness

Focus: Validating value against current tools.

Discussion: We compared the functionality of VirusTotal and MobSF. We found that these tools did not show exported components clearly nor did they handle large files and provide decompiled APK.

Outcome: Described how the tool assists in forensic analysis. The output quality was tested against public standards.

Meeting 7 – User Experience and Web Interface Development

Focus: Transition from standalone script to web-based, accessible utility.

Discussion: Created and developed a responsive web interface with Flask as the backend and Bootstrap for styling. The purpose was to simplify use and to display analysis results in a more straightforward and user-friendly format.

Outcome: We provided a simple drag and drop upload feature with an analysis button so users can upload APKs with ease. Analysis results are shown in an interactive table with counts for API level, permissions, activities, services, and exported/dangerous status. Users can view CSVs directly in the browser or download them individually or all as a ZIP file. The interface provides validation messages, an organised layout, and easy navigation.

Meeting 8 – Real Time Integration and Refining Output

Focus: Web scraping for Integrated Android Developers.

Discussion: Linked tool to scrape the official Manifest. Permission page. I retrieved live permission descriptions and categorised them dynamically at runtime.

Outcome: CSV reports now also contain permission name, description, and if it is dangerous or not. Dangerous permissions are counted and shown cumulatively.