Google Summer of Code 2025 Proposal

Title: Android Virtual Printer Application (for IPP-based Testing and Debugging)

Name: Sauhard Gupta
Email: sauhard74best@gmail.com
GitHub: https://github.com/Sauhard74/Print_it
Location: India

Synopsis

The proposed project aims to develop a fully configurable Android-based Virtual Printer application that acts as a test environment for IPP-compatible print jobs. It can simulate real printer behavior, respond to print jobs, and help developers debug and validate print service functionality across Android and other platforms. This is particularly useful in environments where physical printers are unavailable or impractical to test with.

Benefits to the Community

With a wide range of Android devices and an equally diverse array of printers, it's almost impossible to reproduce every issue that surfaces during printing. This Virtual Printer fills that gap by providing a powerful, flexible, and lightweight emulation tool:

- ChromeOS and Android teams can test printing flows without needing hardware.
- 3P Android app developers can debug user-submitted bug reports.
- QA teams can automate testing in CI pipelines.
- Developers can replicate real-printer behavior using uploaded IPP attribute dumps.
- The open-source community benefits from a reusable, modular emulator.
- Educational use: Students and new developers can explore printer protocols without hardware.
- Integration testing with CI/CD tools for printing workflows.

The application has already been tested on multiple OSes, including Android, Windows, macOS, iOS, and Linux.

Deliverables

Phase 1 (Community Bonding + Initial Development)

- Finalize tech stack, architecture, and module responsibilities
- Detailed design document and API contracts
- Improved documentation of the current implementation
- Mentor feedback integration from the early prototype
- UI/UX refinement for configuration and monitoring

Phase 2 (Core Implementation)

- Implement full support for all major IPP operations (Get-Printer-Attributes, Print-Job, Send-Document, Validate-Job)
- Print Job Simulator: Simulate edge cases like paper jam, low toner,and offline printer
- Upload and parse IPP attribute dumps from real printers to configure capabilities
- Add a robust print job queue with hold/release/cancel support
- Add extensive unit and integration tests
- Role-based job permissions (admin mode for advanced debugging)
- Built-in document viewer for standard formats

Phase 3 (Core Implementation)

- Retrieve IPP Attributes directly from real printers via network (using IPP Protocol) and auto-generate configuration files
- Refactor for modularity and extensibility
- Cloud storage integration (optional)
- Final documentation (user guide, dev guide, architecture)
- Submission and walkthrough demo video
- Exportable print logs for analysis/debugging

Phase 4 (Bonus Deliverables (Beyond Scope, Stretch Goals)

- IPP Job Visualizer: Graphical view of job attributes and timelines
- Custom Plugin Framework: Allow developers to add mock behaviors via scripting
- Performance Benchmarking: Simulate large print queues and concurrency loads
- CI Integration SDK: Provide a module to run tests using the virtual printer in CI pipelines

Timeline

Pre-GSoC Progress (Already Completed)

- Built and deployed a full prototype with modern UI and real-time job tracking
- Implemented basic IPP server using Ktor and JIPP
- Built file format recognition and recovery logic
- Added upload feature for IPP attribute dumps (mentor-requested)
- Received and incorporated feedback from mentor Nathan Muggli

Community Bonding (April 22 - May 19)

- Engage with mentors and the community
- Finalize milestone breakdown and development tools
- Set-up CI pipelines for testing emulator logic

Phase 1 (May 20 - June 16)

- Enhance PrinterService to dynamically load and refresh capabilities
- Add job queue controller (hold/release/cancel)
- Improve DocumentProcessor with internal viewer integration
- Refactor the attribute parser and add real-time validation

Phase 2 (June 17 - July 14)

- Develop a print job simulator for error testing (paper jam, out of paper)
- Build exportable print job logs and trace analysis tools
- Implement a plugin framework prototype for mock behaviors
- Extend UI for job metadata, logs, and control actions

Midterm Evaluation (July 15 - July 19)

- Submit demo, unit tests, performance benchmarks, Mentor feedback, etc

Phase 3 (July 20 - Aug 11)

- Finalize cloud integration and role-based access options
- Add CI SDK module for testing print logic
- Improve UX polish and animations in Compose
- Write comprehensive documentation

Technical Architecture

System Design

1. PrinterService (Core)

- Starts embedded IPP server (Ktor-based)
- Registers as a network printer via Android NSDManager
- Processes IPP requests and manages print job flow

2. DocumentProcessor

- Parses binary data, extracts, or wraps PDFs
- Identifies actual formats (PDF, PNG, JPEG, PS)
- Renames and stores files correctly

3. IPP Handler (via JIPP)

- Handles: Print-Job, Create-Job, Send-Document, Get-Printer-Attributes
- Sends responses in IPP-compliant format

4. Configuration Manager

- Loads IPP attribute dump files
- Dynamically modifies capabilities and printer descriptors

5. UI Module

- Jetpack Compose-based layout
- Settings screen: Modify printer name, load attributes
- Print job viewer: Status, format, timestamp, delete/preview jobs

6. Notification & Recovery

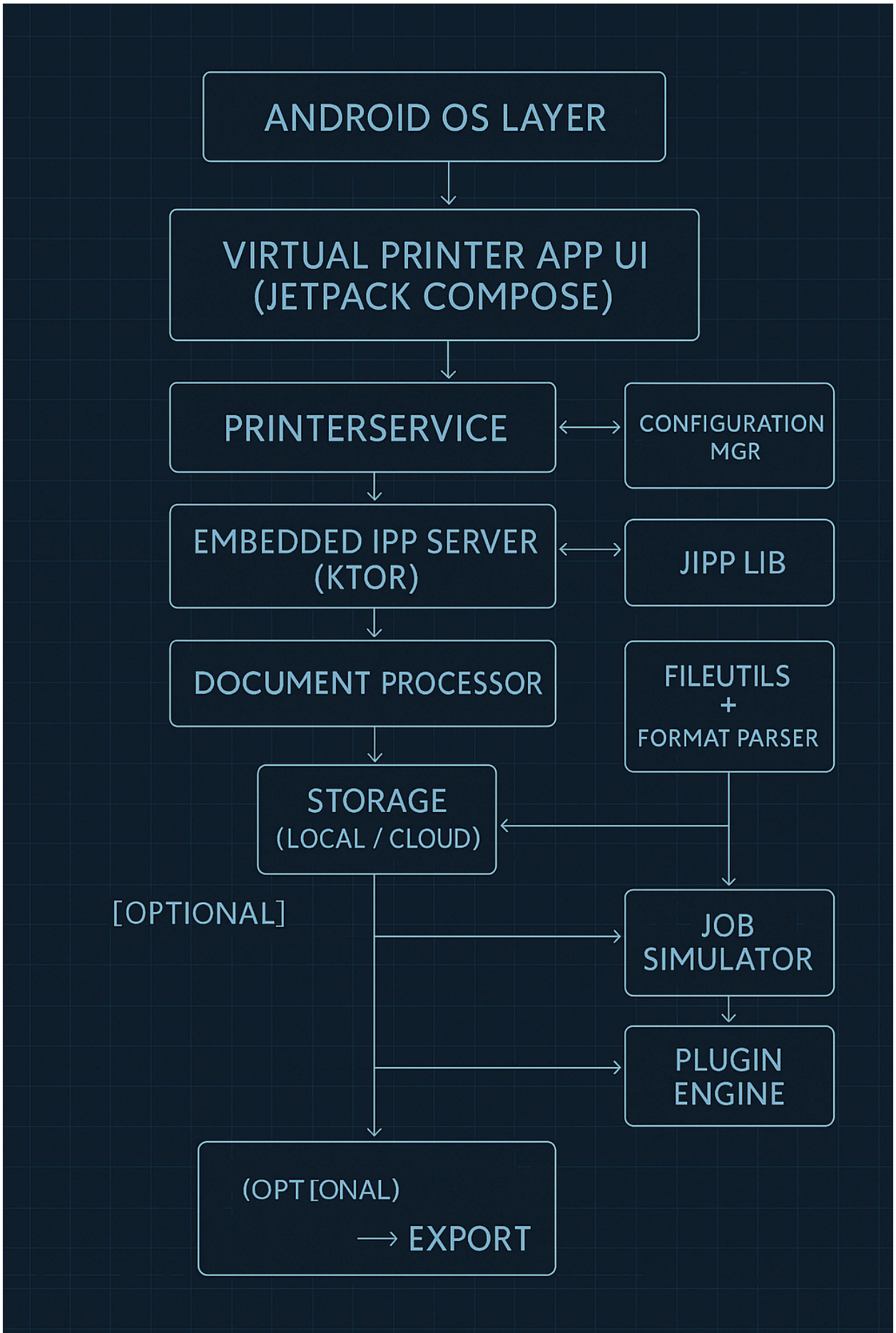
- Uses BroadcastReceiver for live updates
- Auto-recover and renames corrupted files

7. Job Simulator & Plugin Engine (New)

- Simulates error cases for advanced debugging
- Supports custom plugins for response simulation

8. Analytics & Logging System

- Tracks job timelines, metadata, and error cases
- Exports debug logs for analysis or training datasets


ANDROID OS LAYER
VIRTUAL PRINTER APP UI (JETPACK COMPOSE)
PRINTERSERVICE
CONFIGURATION MGR
EMBEDDED IPP SERVER (KTOR)
JIPP LIB
DOCUMENT PROCESSOR
FILEUTILS + FORMAT PARSER
STORAGE (LOCAL / CLOUD)
[OPTIONAL]
JOB SIMULATOR
PLUGIN ENGINE
(OPTIONAL) → EXPORT

Related Work

- **ippeveprinter (PWG)**: Open-source IPP test server, limited to Linux/macOS, C/C++
- **JIPP**: Java-based IPP library used in this project for parsing and composing packets
- **Android PrintService Framework**: Used by OEM apps, not sufficient for built-in printer emulation

This project builds on ideas from `ippeveprinter` and brings them natively to Android in a portable, user-friendly manner.

Experience and Contributions

I've already:

- Built a working Virtual Printer app using Kotlin + Jetpack Compose
- Integrated IPP handling via Ktor and JIPP
- Implemented file format detection and PDF extraction
- Enabled configuration via IPP attribute dumps
- Added real-time job notifications and a modern UI

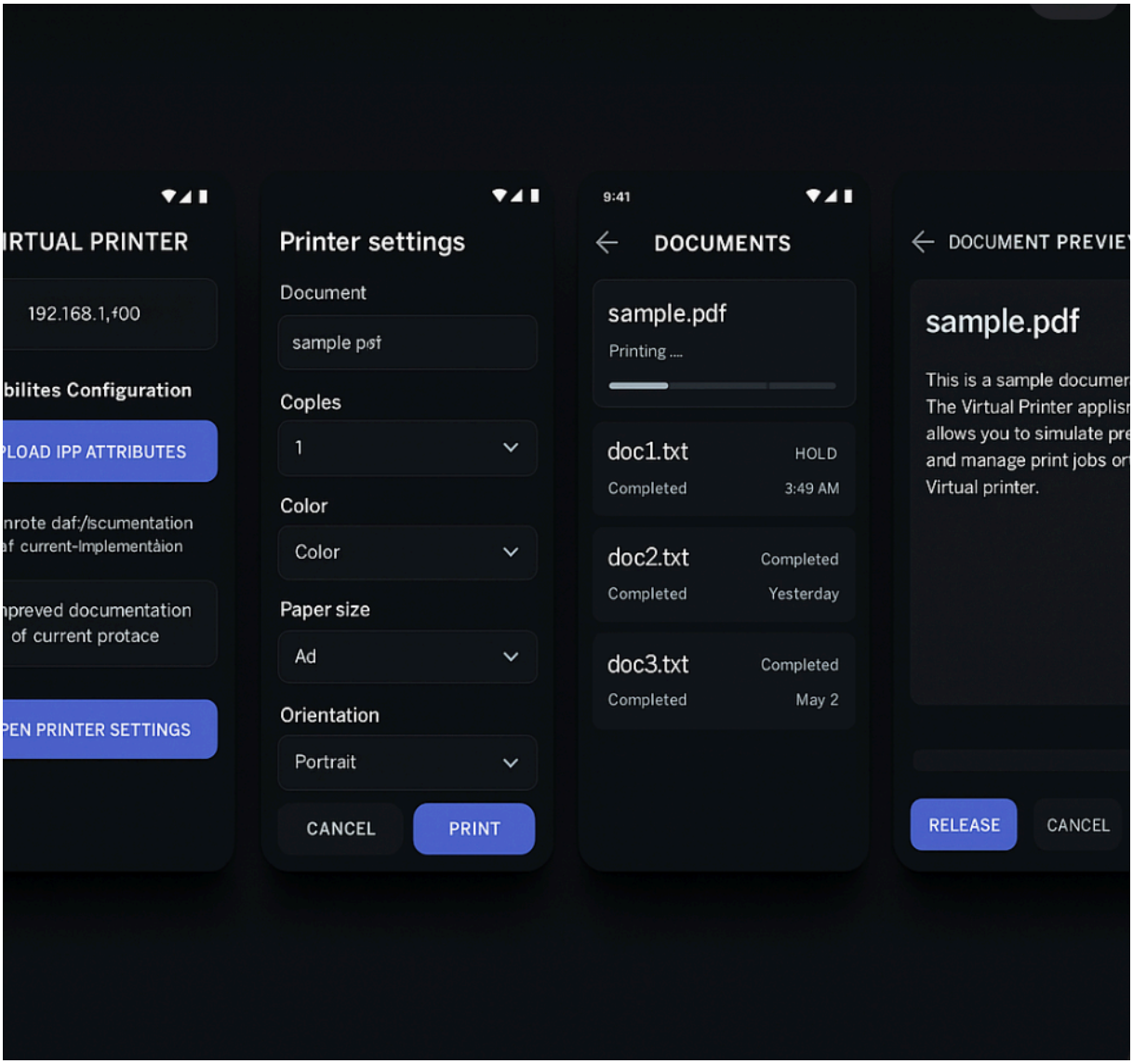
Project Repo: https://github.com/Sauhard74/Print_it
Commits So Far: https://github.com/Sauhard74/Print_it/commits/main/
Working Prototype: https://tinyurl.com/7yz646sp

Mentor Feedback Received: Nathan Muggli reviewed the prototype and suggested configuration support using real printer attribute dumps. I implemented that feedback and pushed improvements accordingly.

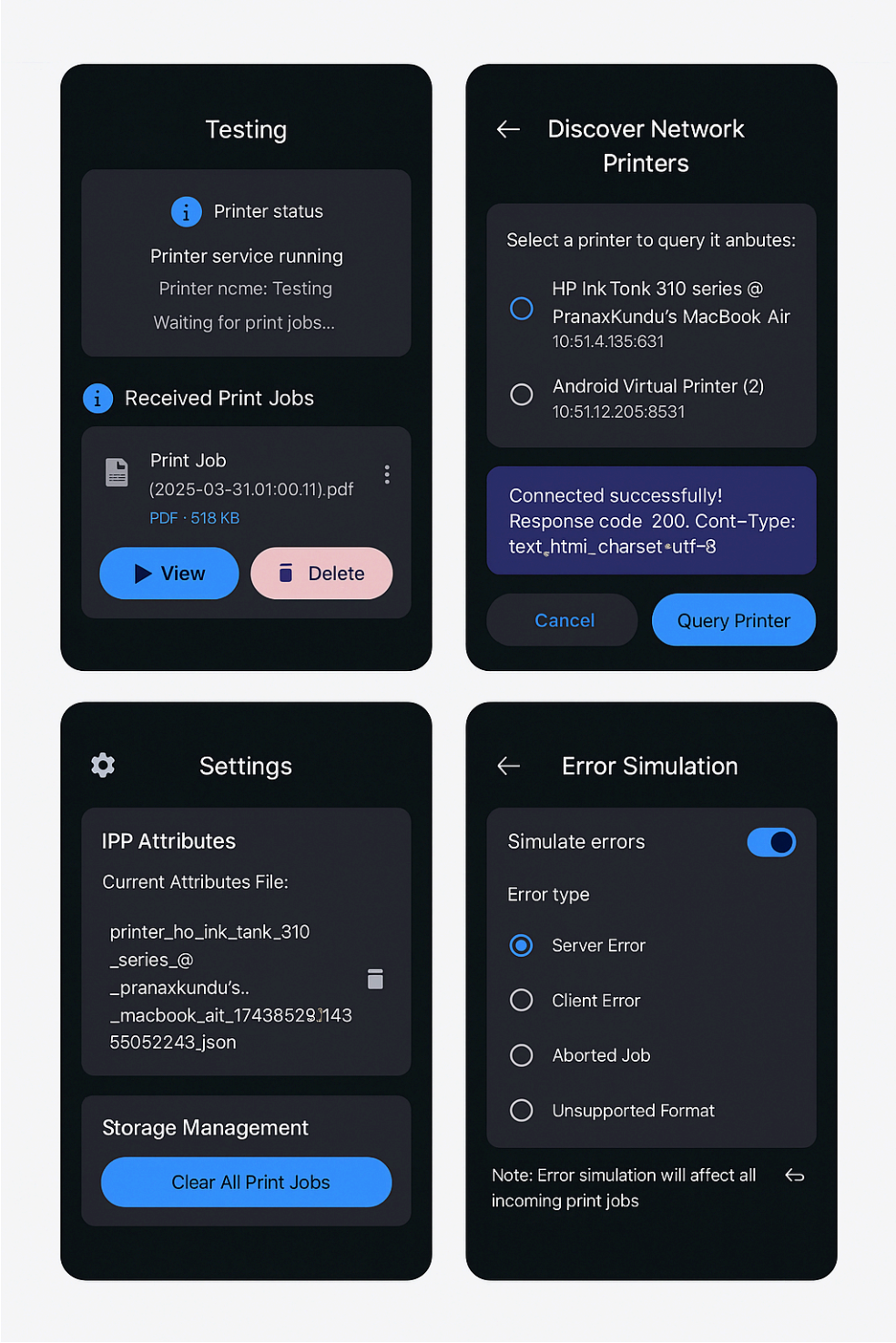
Designs and UI UX Thoughts :

The UI/UX designs we've discussed so far prioritize simplicity, modern aesthetics, and intuitive navigation. The dark mode theme with contrasting blue, gray, and white accents enhances visual appeal while reducing eye strain for users. The clean, minimalist layout ensures that all controls — like print status, network discovery, and settings — are easily accessible, providing a seamless and engaging experience. These designs are optimized for both usability and developer needs, ensuring that the app remains highly functional while offering an enjoyable user interface

1. Figma Designs (Ideation)



2. Technical Details Designs So far:



Personal Background

Name: Sauhard Gupta
University: Birla Institute of Technology and Science Pilani (BITS PILANI)
Program: Bachelor's in Computer Science (Expected 2027)
Location: India

Relevant Skills

- Kotlin, Jetpack Compose, Java, Ktor
- Embedded systems & networking
- Binary protocols (IPP)
- Open source tools & Git workflows

Time Commitment

I will be able to dedicate **25–30 hours per week**, with clear availability throughout the program. I would love to spend much of my time thinking in and out of the project, gaining deep knowledge, and covering all cases rather than just making a prototype.

Final Thoughts

This project combines my passion for Android development and open-source tooling with the practical needs of the ChromeOS and Android printing stack. I've already built a working prototype, received mentor validation, and laid out an ambitious roadmap that pushes beyond expectations.

By incorporating advanced debugging, job simulation, and extensibility through plugins and CI testing, I intend to create a production-grade tool that benefits not only ChromeOS but the broader Android and developer community.

I am already thankful to the mentors for the feedback and guidance.

Thank you for considering my proposal!