# Flashcard Study App — Final Report (Group 05)

Repository: ahmedyero2-prog/group05-flashcard-study-app

Team & Roles (from repo credits) - Project Lead: Dalekee (listed in README credits) - Contributors / Collaboration: ahmedyero2-prog, Stevereeds

## 1) Project summary

The Flashcard Study App is a desktop application implemented in Python with a Tkinter GUI that enables users to create and manage flashcard decks, run study sessions with card flipping, and track progress. Decks are stored locally as JSON files; the app emphasizes a simple, focused study loop and basic progress tracking to support effective revision.

## 2) MVP features delivered (as implemented in the repo)

* Deck Management: Create new decks or load existing decks from the main menu. (See src/models.py, src/persistence.py and data/.)
* Card Creation: Add new flashcards to any deck via the GUI (src/main.py).
* Local Persistence: Decks and progress are saved automatically to data/ as JSON files.
* Study Mode: Study cards in a randomized order; the UI tracks how many cards have been completed and shows basic score tracking.
* Error Handling: The app presents user-friendly pop-ups for corrupted files or missing dependencies to prevent crashes.

These features match the capstone MVP acceptance criteria: create a deck, add cards, run a session, and save progress locally.

## 3) Design & architecture

* Language & runtime: Python 3.11 (README states Python 3.11). Pipenv or a virtual environment is recommended.
* Package layout (as in repo):
  + Top-level: Pipfile, Pipfile.lock, README.md, deck\_import\_guide.txt, data/, tests/, .github/workflows/.
  + Source code lives in src/ and contains: main.py, models.py, persistence.py, exceptions.py (per README file listing).
* Core modules:
  + models.py — Domain models (Card, Deck, Session).
  + persistence.py — Save/load logic for JSON deck files.
  + main.py — Application entrypoint and GUI bootstrap.
* Dependencies noted in README: Tkinter (GUI), Pillow (image handling), Pytest (tests). The README also notes automated dependency checks on first run.

## 4) Tests & CI (current state from repo)

* Test files: tests/test\_core.py exists and is invoked in the README with pytest --capture=no tests/test\_core.py.
* Local test instructions from README: Running that command should print All tests passed successfully! if passing.
* CI: The repo contains a .github/workflows folder; if a CI workflow is present it should run tests on push — otherwise adding the provided CI template is recommended to ensure reproducible grading.

## 5) How to run (copied & adapted from README)

1. Clone the repository:

* git clone https://github.com/ahmedyero2-prog/group05-flashcard-study-app.git  
  cd group05-flashcard-study-app

1. Create and activate a virtual environment (recommended):

* python -m venv venv  
  # Windows  
  venv/Scripts/activate  
  # macOS / Linux  
  source venv/bin/activate

1. Start the application from the src/ folder:

* cd src  
  python main.py

1. Run tests from the project root (as suggested in README):

* pytest --capture=no tests/test\_core.py

Note: If your environment uses pipenv or Pipfile, run pipenv install --dev then pipenv run pytest or pipenv run python src/main.py as appropriate.

## 6) Challenges & known issues (repo-documented + additional notes)

* The README documents the run and test commands clearly. Common issues students encounter that you should double-check before submission:
  + Test import path issues: Ensure tests import the modules from src/ correctly. If pytest cannot find src, use PYTHONPATH=./src pytest or add pytest.ini with pythonpath = src.
  + Demo & report artifacts: The README mentions features and run steps, but confirm that a demo video link (demo.mp4 or YouTube) and the final report PDF are included in the repo before tagging v1.0.
  + Dependency differences: The README claims the app will automatically check/install dependencies. Make sure this step is reproducible on the grader’s machine or document manual pip install steps.

## 7) Future work / stretch goals

* Implement a formal spaced-repetition scheduler (SM-2 or variant).
* Add import/export deck functionality (Anki-compatible or JSON templates). The repo already includes a deck\_import\_guide.txt which will help this feature.
* Improve UI with animated card flips and progress visualizations.
* Harden tests and add CI to run pytest on push.

## 8) Artifacts in the repo (verified)

* Pipfile, Pipfile.lock
* README.md (contains run/test instructions)
* deck\_import\_guide.txt
* src/ (source files: main.py, models.py, persistence.py, exceptions.py per README listing)
* data/ (storage for saved decks)
* tests/test\_core.py
* .github/workflows/ (CI folder — verify workflow file exists)

## 9) Final notes & recommended next steps before submission

1. DEMO video in (https://drive.google.com/file/d/1n6Dc2xPpCRNAQLZetbwGxdkFxQSU3Xl9/view?usp=drivesdk)
2. Add the final report PDF (this document) to repo root or docs/ and include 200-word reflections from each team member.
3. Ensure tests run in CI: add or verify .github/workflows/ci.yml runs pipenv install --dev and pytest.
4. Double-check roles & commit history: update README Credits to include full names and one-line contributions, and make sure commit history supports the roles.

*Prepared using the repository contents and README (as of last access). Replace any placeholder names with real student names and include demo/report artifacts prior to final submission.*