GlassGuard Project Report

**Period:** July 9 – July 21 2025  
**Prepared by:** Alok Kulkarni  
**Reviewed by:** Jonathan Bozanin (General Manager)

## 1. Purpose

On July 9, Jonathan Bozanin asked us to improve our data collection and reporting system so that every scratch on custom glass runs is recorded with:

* Where on the panel the scratch appeared
* What kind of scratch it was
* Which glass coating was used
* What rack carried the glass (A‑frame, bungee cart, etc.)
* A photograph of the scratch

Our aim was to deliver a single web‐based tool where operators can enter scratch data, see real time summaries, and download reports and images.

## 2. Tools and Data Storage

* **User Interface:** Streamlit (a Python‑based web app framework)
* **Data Storage:** SQLite (a simple file‑based structured query language database)
* **Charts:** Plotly Express (for bar, pie, and line charts)
* **Images:** Stored directly in the database as raw bytes (binary large object, or “BLOB”)
* **Version Control & Deployment:** GitHub and Streamlit Cloud

## 3. What We Built

# A. Dashboard Tab

1. **Year Selector** – choose which calendar year’s data to view
2. **Core Defect Summary** – bar chart showing total count of each scratch type
3. **Breakdown by Glass Coating** – choose one scratch type and see counts by coating
4. **Weekly Scratch Volume** – line chart of total scratches per week
5. **Glass Type, Rack Type, and Vendor Charts** – pie charts showing distribution

# B. Data Entry Tab

1. **Input Fields:**
   1. Purchase Order Number
   2. Tag Number
   3. Glass Size
   4. Quantity
   5. Date of record
   6. Scratch Location (e.g. top‑left, center)
   7. Scratch Type (e.g. scratch, production issue)
   8. Glass Coating Type (e.g. Clear, Lowe272)
   9. Rack Type (A‑frame, bungee cart)
   10. Vendor
   11. Notes
2. **Photo Upload:** Accepts JPG/PNG up to 2 MB, shows a small live preview
3. **Save Action:** Stores all fields plus the image bytes into the SQLite database

# C. Data Table Tab

1. **Summary Metrics:**
   * Total records logged
   * Total scratches recorded
   * Glass coating with the highest scratch count
2. **All Scratch Records Table:**
   * Columns: Tag, Date, Quantity, Scratch Type, Glass Coating
   * Export to Excel spreadsheet
3. **Image Preview & Download:**
   * Select any Tag number to view a small preview of the scratch photo
   * Click to download the original image file

**4. Step‑by‑Step Timeline**

|  |  |
| --- | --- |
| Date | Work Completed |
| Jul 9 | Met with General Manager Jonathan Bozanin to record new requirements: scratch details plus photo |
| Jul 10 | Created the SQLite database file and the Streamlit project structure |
| Jul 11 | Built the Dashboard tab with year selector and basic charts |
| Jul 14 | Designed the Data Entry form: fields, validation, and save logic |
| Jul 15 | Added the “All Scratch Records” table and Excel export feature |
| Jul 16 | Implemented photo upload with instant preview in the form |
| Jul 17 | Moved photo storage into the database as raw bytes (BLOB column) |
| Jul 18 | Enabled photo retrieval in the Data Table; handled missing dates and cleaned up test data |
| Jul 21 | Final testing, user‑interface tweaks, deployment setup, and documentation |

**5. Challenges and How We Solved Them**

* **Empty or Invalid Dates:** Converted all date text to real dates; unknowns show as “unknown.”
* **Photo Size Limits:** Enforced 2 MB maximum; larger images are rejected.
* **Test Data Cleanup:** The app detects any Tag containing “test” and deletes it immediately on refresh—test entries show up once (for preview) then vanish on reload.
* **Image Storage Method:** Started with saving hex files, then simplified by storing raw bytes in the database.

**6. Next Steps**

1. Automated Backups

* Goal: Every night, make a copy of the main database (glass\_defects.db) and store it safely online (for example, in a cloud folder or on GitHub).
* How:
  + Option 1: Use GitHub’s automation tools to upload a backup every day.
  + Option 2: Add a reminder inside the app to run a backup script each night.

1. User Access Control

* Goal: Only authorized staff (for example, Supervisors or Admins) should be able to delete or change records.
* How:
  + Add a login screen (username and password) to unlock sensitive features like deleting records.
  + If the company already has a Single Sign-On (SSO) system, we can connect the app to it.

1. Production-Line Integration

* Goal: Allow operators to enter rejected glass records right from the production line using a tablet or computer, instead of writing on paper and entering later.
* How:
  + Host the app inside the company’s network or on a secure server.
  + Add a “Quick Entry” mode with bigger buttons and faster data entry for shop-floor use.

**End of Report**  
July 21 2025