Building AI Application Challenge 2026 Builder Progress Log & Submission Workbook

Days 2–8 (Environment → Build → Evaluate → Deploy → Submit)

Use this document as a living workbook. Update it daily with links to your repo, demos, screenshots, and decisions.

0) Participant & Project Metadata

Participant Name	Bhadmasree Aanantharaman
Email	22501000083@lu.ac.ae
Timezone	GST (Dubai)
Start Date	
Path (select one)	☑ ~~☐ Airia~~ ☐ ☐ LLM/API Integration ☐ ☐ No-Code/Low-Code
Project Repo URL (GitHub)	https://github.com/bhadmasreea20/braille-bridge-ai/blob/main/README.md
Demo / App URL (if available)	

Quick links (keep updated):

- ☑ ~~☐ GitHub repo (public link)~~ https://github.com/bhadmasreea20/braille-bridge-ai
- ☐ ☐ Deployment link (Hugging Face / Streamlit / other)
- ☐ ☐ Demo video link (Loom/YouTube)
- ☐ ☐ LinkedIn progress post URL
- ☐ ☐ Tweet/X progress post URL (if applicable)

3) Data / Knowledge Sources Inventory

Day 2 focus: list your data sources clearly. This becomes critical for quality, evaluation, and judging.

Source	Type	Owner / License	Format	Size	Update freq	Access method	Preprocessing needed	PII/Sensitive?	Status
Structured Braille Character mapping	Reference dataset	Public educational sources	Text/structured table			Manual upload	Formatting & validation	no	planned
Sample accessibility use cases	research	public	notes			Manual entry	none	no	planned

If you are doing RAG / search:

☐ Corpus assembled (what content?)
☐ Chunking strategy defined (size/overlap)
☐ Embedding model selected
☐ Vector store selected (or file-based retrieval)
☐ Citation strategy (how will you show sources?)
☐ Data safety: remove secrets/PII and respect copyright

4) LLM / Model Selection & Experiments

Capture what you tried, what worked, and why you chose your final stack.

Candidate	Provider	Why considered	Prompting approach	Quality notes	Latency notes	Cost notes	Decision
GPT-4o	OpenAI	High-quality multimodal	Structured	Very accurate	Moderate	Expensive	Not selected due to cost
Gemini 2.0 Flash-Lite	Google	Low cost + vision support	Structured translation-only prompt	Good with clear images	Fast	Very affordable	Selected

Prompt versions (for LLM/API or No-Code tools):

Prompt ID	Goal	System constraints	Few-shot examples?	Tools/Actions used	Notes / Results
V1	Translate Braille image to English	Output only translation, no explanation	No	Vision processing	Worked but sometimes added extra text
V2	Improve output cleanliness	Return ONLY translated English text. If no Braille, respond: "No Braille detected."	No	Vision	Improved reliability

Day 1 – Project Proposal & Planning

On Day 1, the project idea for Braille Bridge was conceptualized and defined. The problem identified was the communication gap between visually impaired individuals who use Braille and those who cannot read it. The proposed solution was an AI-powered application capable of translating Braille from images into readable English text.The initial technical plan included:
Using a multimodal AI model capable of processing images.

Building a simple frontend interface for image upload and output display.

Using GitHub for version control and documentation.

The project scope, target users, and expected functionality were clearly outlined.

5) Day 2 — Environment Setup & Initial Development

Deliverable for Day 2: no full project submission today—prepare your environment + tech stack/tooling + initial work evidence.

Environment setup checklist (tick what applies):

- ☑ ~~☐ Repository created and initial commit pushed~~
- ☑ ~~☐ README created (project goal + how to run)~~
- ☐ ☐ .gitignore configured (no secrets)
- ☑ ~~☐ Python environment created (venv/conda) OR platform workspace created (no-code)~~
- ☐ ☐ Dependencies installed and pinned (requirements.txt / pyproject / lockfile)
- ☐ ☐ API keys stored safely (.env, secrets manager, or platform secrets)
- ☐ ☐ Basic 'hello world' run completed locally
- ☑ ~~☐ Basic API call tested (if using LLM/API) OR first workflow run (if no-code)~~
- ☐ ☐ Folder structure created (src/, data/, notebooks/, etc.)
- ☐ ☐ First prototype screen/flow created (UI stub acceptable)

Evidence (links):

GitHub repo URL	https://github.com/bhadmasreea20/braille-bridge-ai
Commit hash / tag	
Screenshot(s) link	
Notes on setup decisions	

Initial development log (what you built today):

• Finalized project concept and scope

• Defined target users and key features

• Created GitHub repository with initial documentation

• Set up Airia workspace

• Drafted initial application workflow

Blockers / issues encountered (and how you resolved them):

• Exploring best method to structure AI workflow inside Airia

• Need to define evaluation strategy for translation accuracy

Day 2 – System Design & Architecture Setup

On Day 2, the system architecture was designed. The workflow was structured as:

Image Upload → Multimodal AI Model → English Text Output

The GitHub repository was created and structured properly with README documentation. Basic project folders were organized, and security practices such as excluding API keys were implemented.

Potential risks were identified, including:

Image clarity issues

Incorrect Braille detection

API cost limitations

Mitigation strategies were defined to address these risks.

Day 3 — Building the Brain of Your App

Checklist:

- ☑ ~~☐ Core logic implemented (model/prompt/workflow)~~
- ☑ ~~☐ Data ingestion or API integration expanded beyond hello world~~
- ☑ ~~☐ 10–20 test questions/examples drafted (start your 'exam set')~~
- ☑ ~~☐ First measurable baseline created (even if rough)~~
- ☑ ~~☐ README updated with run instructions~~

Artifacts / notes:

What is the 'brain' of your app (1–2 sentences)?	The brain of my application is an AI-powered translation engine using the OpenAI GPT-4o model. It processes Braille input (text/image) and generates readable English output through a structured API workflow integrated into my no-code application.
Link to key code/workflow	Implemented inside Bubble workflow using OpenAI Chat Completion API integration.
Baseline results (short)	Initial tests using sample Braille text showed successful translation output when structured prompts were used. Text-based Braille input was successfully processed and returned accurate English translation. Vision-based image handling is partially functional and under refinement.
What you will improve next	Improve image processing reliability • Add loading states and better error handling • Expand test dataset to measure translation accuracy • Implement audio playback for accessibility

Decisions made today (why):

• Selected GPT-4o model due to multimodal capability (text + image support).

• Used structured prompt format to ensure clean translation output without extra explanation.

• Simplified workflow logic to ensure reliable API response mapping before adding advanced features.

Blockers / help needed:

• Optimizing image input formatting for consistent translation results

• Defining a formal evaluation metric for translation accuracy

Day 3 – Core Implementation (Building the Brain)

Day 3 focused on implementing the core functionality using Airia. A cost-efficient multimodal vision model was selected to balance performance and budget.

Key activities included:

Creating the AI agent inside Airia.

Configuring image input and text output.

Designing a structured system prompt to ensure translation-only output.

Connecting Input → Model → Output in the pipeline.

Testing initial Braille image samples.

Updating the GitHub repository with the working prototype.

By the end of Day 3, a functional image-to-text translation system was operational.

Day 4 — Optimizing Integration & Application Evaluation

Checklist:

- ☑ ~~☐ Evaluation approach defined (metrics + test set)~~
- ☑ ~~☐ Error handling + retries added (API) OR validation rules added (no-code)~~
- ☑ ~~☐ Prompt/model iteration based on failures~~
- ☑ ~~☐ Latency/cost notes captured~~
- ☑ ~~☐ Safety/guardrails considered (content, PII, injection)~~

Artifacts / notes:

Evaluation dataset link / location	Self-curated test image folder (local testing set)
Metrics used (accuracy, faithfulness, etc.)	Translation accuracy Output cleanliness Latency Cost per request
Top failure modes found	Blurry images reduce accuracy Non-Braille patterns cause hallucination
Fixes applied	Strengthened prompt constraints Added rejection response Reduced randomness

Decisions made today (why):

- Finalized the use of a cost-efficient multimodal vision model (Gemini 2.0 Flash-Lite) to balance performance and budget.
- Strengthened the system prompt to enforce translation-only output.
- Added fallback responses for unclear or non-Braille images.
- Defined evaluation metrics (accuracy, latency, cost).
- Decided not to implement RAG since the application does not require retrieval.

Blockers / help needed:

- Image clarity affects translation accuracy.
- Cost constraints limit experimentation with larger models.
- Need larger evaluation dataset for more reliable benchmarking.

Day 4 – Evaluation, Optimization & Refinement

Day 4 focused on testing and improving the system. A small curated dataset of Braille images was created for evaluation, including clear images, slightly blurred images, and non-Braille images for negative testing.

Evaluation metrics defined:

Translation accuracy

Output cleanliness (no extra explanations)

Response latency

Cost efficiency

Based on testing results:

The system prompt was strengthened to enforce strict output rules.

Fallback responses were added for unclear or non-Braille images.

Model randomness was reduced for consistent results.

Cost strategy was finalized using a lightweight multimodal model.

By the end of Day 4, Braille Bridge had a stable functional prototype with defined evaluation methodology, documented improvements, and optimization strategies.

Day 5 — Integration of Model/API with Interface

Checklist:

- ☑ ~~☐ User flow designed (screens + inputs/outputs)~~
- ☑ ~~☐ Interface connected to backend/logic~~
- ☑ ~~☐ UX basics: loading, error states, reset/clear~~
- ☑ ~~☐ Logging of inputs/outputs enabled (safe logging)~~
- ☑ ~~☐ Demo link created (even if rough)~~

Artifacts / notes:

Interface tech (Gradio/Streamlit/React/No-code UI)	Base44 (No-code UI platform)
Demo link	Base44 Demo: https://braille-bridge-audio.base44.app Airia Brain: https://airia.ai/catalog/chat/965df3ba-3d3c-493a-b791-41c384a41869?deploymentId=26afcb23-c2f2-48d2-9ffc-33af2ab3accc
Screenshots link	https://github.com/bhadmasreea20/braille-bridge-ai/tree/main/images
Known UX issues to fix	Slight delay while processing large images

Decisions made today (why):

- Chose Airia for AI brain due to cost control and multimodal support.
- Used Base44 for faster frontend prototyping instead of full custom code.

Blockers / help needed:

- Improving accuracy for blurry Braille images

Day 5 Progress Report

Day 5 focused on full system integration and usability refinement. The Base44 frontend was successfully connected to the Airia AI brain, enabling a complete end-to-end workflow from image upload to translated text output.

Comprehensive testing was conducted using curated Braille images, blurred samples, and non-Braille inputs to evaluate system robustness. Prompt constraints were strengthened, and fallback responses were improved to reduce hallucinated outputs.

UI enhancements were implemented, including clearer instructions, loading indicators, and improved output formatting. By the end of Day 5, the application achieved stable end-to-end functionality with improved reliability and user experience.

Day 6 — Final Enhancements, Security & Debugging

Checklist:

- [x] ~~Input validation + sanitization~~
- [x] ~~Secrets handling reviewed (no keys in repo)~~
- [x] ~~Rate limits / caching considered~~
- [x] ~~Bug list triaged and reduced~~
- [x] ~~README + architecture notes cleaned up~~

Artifacts / notes:

Security checklist notes	No API keys exposed in frontend code Airia handles model execution securely No user data stored permanently Images processed in-session only No external data retrieval (no RAG used)
Top bugs fixed	• Fixed incorrect formatting of translated output • Improved error message when no Braille detected • Reduced hallucinated responses by strengthening system prompt
Remaining risks	Reduced performance on low-quality images Model may misinterpret embossed patterns as Braille Needs real-world dataset testing

Decisions made today (why):

- Strengthened system prompt to return only translated text to reduce hallucinations.
- Focused on stability over adding new features.

Blockers / help needed:

- Need curated real Braille dataset for better validation.
- Exploring cost optimization for scaling usage

Day 6 Progress Report

Day 6 focused on validation, documentation, and deployment planning. A structured evaluation was conducted using curated Braille image samples and defined performance metrics including accuracy, response time, and cost efficiency.

Safety and responsible AI considerations were documented, ensuring no personal data processing and enforcing strict translation-only output through prompt guardrails.

The system architecture is now stable, with the Airia AI brain handling vision-based translation and the Base44 frontend delivering a functional user interface. Deployment readiness and future scaling strategies have been outlined, positioning Braille Bridge for further refinement and potential real-world use.

Day 7 — Final Review & Deployment

Checklist:

- ☑ ~~☐ Deployment target chosen and deployed~~
- ☑ ~~☐ Environment variables set in deployment platform~~
- ☑ ~~☐ Smoke tests run on deployed version~~
- ☑ ~~☐ Performance checked (latency/cost)~~
- ☑ ~~☐ Submission package checklist started~~

Artifacts / notes:

Deployment platform	Base44 (Frontend) + Airia (AI backend)
Deployment URL	https://braille-bridge-audio.base44.app
Smoke test results	• Image upload tested successfully • Braille image processed correctly • English translation displayed clearly • No crashes during testing • API connection stable
Fallback plan if deployment breaks	• Use local development version • Share recorded demo video • Provide GitHub repository as backup reference

Decisions made today (why):

- Chose Base44 for fast and simple frontend deployment
- Continued using Airia for cost-effective multimodal AI processing
- Optimized prompt to return only translated text to reduce token cost

Blockers / help needed:

- Limited real Braille image dataset for large-scale testing
- Managing API cost while testing multiple iterations

Day 7 Progress Report

On Day 7, the focus was on finalizing the deployment and ensuring that the Braille Bridge application was stable and fully functional. The frontend prototype built using Base44 was reviewed and tested thoroughly. The AI “brain” hosted on Airia was successfully connected and verified to process image inputs and return accurate Braille-to-English translations.

Deployment was completed using the Base44 public hosting link. Environment variables and API connections were checked to ensure secure and stable integration with Airia. Multiple smoke tests were conducted, including image upload testing, API response validation, and UI clarity checks.

Performance was evaluated in terms of latency and cost efficiency. To optimize token usage and reduce unnecessary output costs, the system prompt was refined to ensure that the AI returns only the translated English text without explanations.

Overall, Day 7 ensured that the application moved from the development stage to a deployable, test-ready product.

Day 8 — Final Submission & LinkedIn Sharing

Checklist:

- ☑ ~~☐ Final app link working~~
- ☑ ~~☐ Final repo is clean + documented~~
- ☑ ~~☐ Demo video recorded~~
- ☑ ~~☐ Final submission form completed~~
- ☑ ~~☐ LinkedIn post published + link shared~~

Artifacts / notes:

Final app URL	https://braille-bridge-audio.base44.app
Final repo URL	https://github.com/bhadmasreea20/braille-bridge-ai/tree/main
Demo video URL	Bhadasree Anantharaman-Braille Bri…
LinkedIn post URL	https://www.linkedin.com/feed/update/urn:li:activity:7425766339665031168?utm_source=share&utm_medium=member_desktop&rcm=ACoAAGETyUwBrsTKp8lt7bOti8lMM_iHAReCw8g
Tweet/X URL (if applicable)	NA
Reflection: what you learned	Through this 8-Day AI Application Challenge, I learned how to design, build, integrate, test, and deploy a complete AI-powered application from scratch.

Decisions made today (why):

- Finalized deployment using Base44 for simplicity
- Cleaned GitHub repository for professional submission
- Documented architecture and workflow clearly

Blockers / help needed:

- Scaling the app for larger dataset usage
- Future improvements: better Braille detection accuracy

Day 8 Progress Report

Day 8 focused on preparing the final submission and polishing the overall project presentation. The GitHub repository was cleaned and organized with proper documentation, including a clear project description, tech stack details, architecture explanation, and setup instructions.

A demo video was recorded to demonstrate the full workflow of Braille Bridge — from uploading a Braille image to receiving translated English text. The final deployment link, repository link, and supporting materials were compiled for submission.

Additionally, a LinkedIn post was published summarizing the journey, learnings, and project impact. Reflection on the challenge highlighted key learnings such as multimodal AI integration, prompt engineering, cost management, frontend-backend connectivity, debugging, and deployment practices.

By the end of Day 8, Braille Bridge was successfully completed as a working AI-powered application with a deployed frontend, functional AI backend, documented codebase, and public demonstration.

10) Community Sharing Tracker (bonus points)

Use this to track your social proof and claim points.

Date	Platform	Post URL	What you shared	Tagged DDS?	Points claimed
04/02	LinkedIn	https://www.linkedin.com/posts/bhadmareea20_aiexplorer-aidemos-aiusecases-activity-7424506373993984000-aEqP?utm_source=share&utm_medium=member_desktop&rcm=ACoAAGETyUwBrsTKp8lt7bOti8lMM_iHAReCw8g	AI Explorer ⚡ AI Demos, AI Use Cases and Q & A	Yes	
03/02	LinkedIn	https://www.linkedin.com/feed/update/urn:li:activity:7424333579612164096?utm_source=share&utm_medium=member_desktop&rcm=ACoAAGETyUwBrsTKp8lt7bOti8lMM_iHAReCw8g	Building Braille Bridge – Day 1 to Day 4 Progress	Yes	
01/02	LinkedIn	https://www.linkedin.com/posts/bhadmareea20_ai-accessibility-innovation-activity-7423633924603670529-PVZm?utm_source=share&utm_medium=member_desktop&rcm=ACoAAGETyUwBrsTKp8lt7bOti8lMM_iHAReCw8g	Developing an AI-driven accessibility solution as part of the AI Application Building Challenge.	Yes	
07/02	LinkedIn	https://www.linkedin.com/feed/update/urn:li:activity:7425766339665031168?utm_source=share&utm_medium=member_desktop&rcm=ACoAAGETyUwBrsTKp8lt7bOti8lMM_iHAReCw8g	final project — Braille Bridge	Yes	

11) Reference Resources (from the challenge)

Core challenge walkthrough video: https://www.youtube.com/watch?v=X4PitcxNDjE

Python environment setup: https://www.youtube.com/watch?v=D5XyQ96EgiM&t=759s

GitHub basics: https://youtu.be/bV_9mr5O5bg?si=0oiiT9BZutcKCjnL

OpenAI API key setup: https://www.youtube.com/watch?v=CVnTzj-qhCU&t=8s

Model/LLM selection blogs:

https://decodingdatascience.com/choosing-the-right-gemini-model-for-your-ai-project-a-beginner-friendly-guide/

https://decodingdatascience.com/openai-guide-to-use-which-model-for-tasks/

https://decodingdatascience.com/how-to-choose-the-right-openai-model-gpt-5-complete-guide/

AI Explorer RSVP:
https://nas.io/artificialintelligence/events/ai-explorer-ai-demos-ai-use-cases-and-q-a-1767887350300

12) Reviewer / Mentor Notes (optional)

Use this section if an instructor/mentor is reviewing your progress.

Strengths observed:

-

Areas to improve next:

-

Action items:

-