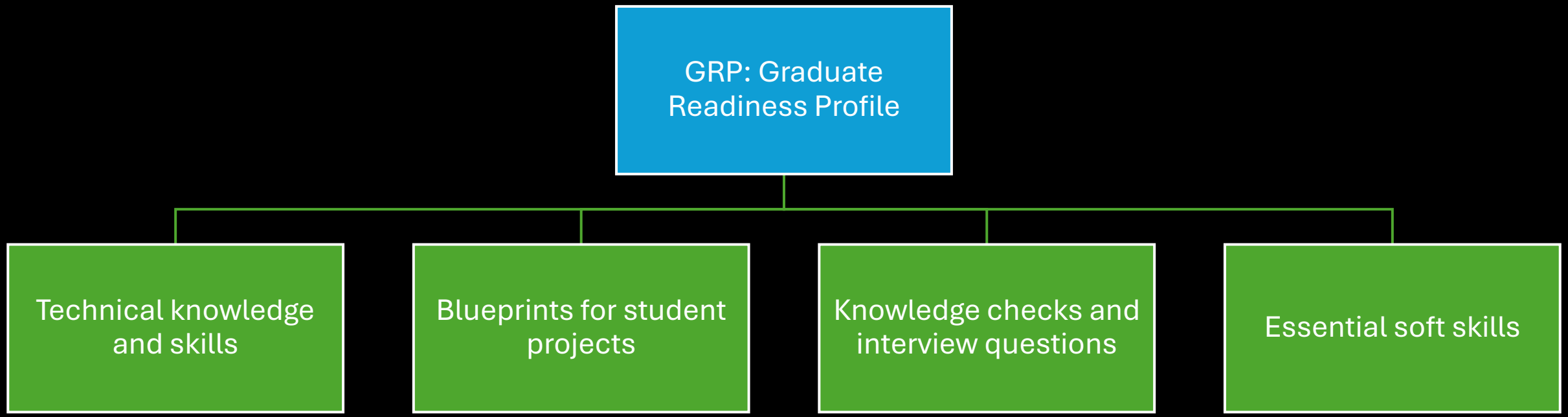
מכון טכנולוגי חולון
Holon Institute of Technology

GRP: Graduate Readiness Profile

Job Role: Full Stack Developer

Alexander(Sasha) Apartsin

What must graduates show to pass entry-level job interviews?

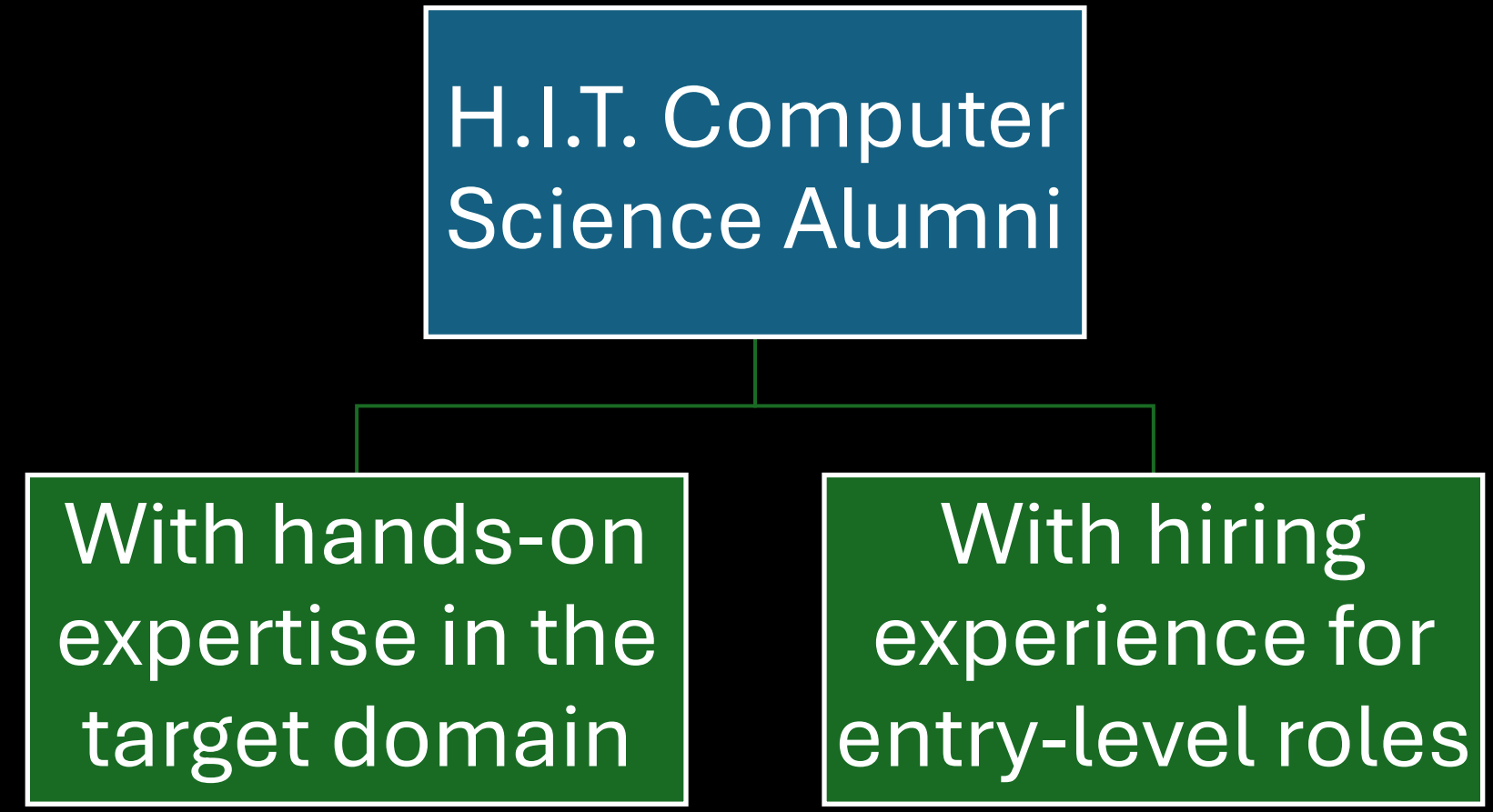


Objectives

Faculty: help shape the curriculum to ensure graduate readiness

Students: help plan studies and choose courses for confident job market entry

GRP Industry Advisory Board(IAB)



Full-Stack Developer: IAB Members

- Software Engineer @**Amazon**
- Expert in cloud technologies
- H.I.T. CS Class of 2014

Boaz Berman

- Software Engineer @**Checkpoint**
- Expert in full-stack development
- H.I.T. CS Class of 2019

Ayelet Avraham

- Senior Software Developer @**Microsoft**
- Expert in full-stack development
- H.I.T. CS Class of 2015

Igor Shor

The launch of the industry advisory board was made possible thanks to Libat Avishai, Head of Alumni & Friends Engagement at H.I.T.

I. Technical Knowledge and Skills

Core Knowledge Domains

#	Domain	Description
1	**Modern dev practices**	This domain focuses on practices and tools for high-quality, maintainable software, including testing, clean code, agile methods, code reviews, and AI-assisted development.
2	**Backend technologies**	Back-end technologies cover the tools, frameworks, and systems used to build and maintain the server-side of applications, which handle data processing, business logic, and communication between the front-end and databases or other services
3	**Frontend technologies**	Front-end technologies encompass the tools, frameworks, and practices used to build the user-facing layer of applications, which is the part users interact with directly
4	**Cloud and DevOps**	The Cloud and DevOps domain encompasses the technologies, tools, skills, and practices required to deploy, manage, and optimize applications in modern cloud environments, while enabling seamless integration between development and operations.

1. Modern Development Practices

Skill	Description	Tools	Use Case	Interview Question
Version Control	Proficiency with Git, branching strategies, merge conflicts, pull requests	Git, GitHub, GitLab, Bitbucket	Collaborating on code with a team and managing code versions	How do you resolve a merge conflict in Git?
Testing	Understanding of unit testing, integration testing, end-to-end testing, and test automation	Jest, Mocha, Selenium, Cypress, TestimIO	Ensuring software quality and reliability through automated tests	Explain the difference between unit tests and integration tests
Clean Code	Principles of writing clean, maintainable code, including naming conventions, SOLID principles, and refactoring	Linters (e.g., ESLint), Code formatters (e.g., Prettier) - generalize	Writing code that is easy to understand, modify, and extend	What does the Single Responsibility Principle mean in software design?
Agile Methodologies	Understanding of Agile principles, Scrum, Kanban, sprint planning, and retrospectives	Jira, Trello, Asana	Working effectively in a team to deliver software incrementally	What is the role of a Scrum Master in an Agile team?
Code Review	Best practices for reviewing code, providing constructive feedback, and using code review tools	GitHub Pull Requests, Gerrit, Phabricator	Improving code quality and sharing knowledge within the team	What do you look for when reviewing someone else's code?

2. Frontend technologies

Skill	Description	Tools	Use Case	Interview Question
JavaScript Fundamentals	Mastery of JavaScript (ES6+), closures, asynchronous programming, promises, async/await	Chrome DevTools, Node.js	Building dynamic and interactive web applications	Explain how closures work in JavaScript
Frontend Frameworks	Proficiency in React (Best; Hooks), Vue.js, or Angular for building user interfaces	React, Vue.js, Angular	Creating reusable components and managing state	What is the virtual DOM in React and how does it improve performance?
Responsive Design	CSS frameworks (Bootstrap, Tailwind), media queries, css specifity, flexbox, grid layouts	Tailwind CSS - investigate	Ensuring web applications work on various devices and screen sizes	How do you make a website responsive?
Browser APIs	Understanding of DOM manipulation, localStorage, Fetch API, WebSockets	Chrome DevTools, Postman	Interacting with browser features and external services	How do you handle cross-origin requests in a web application?
State Management	Managing application state using Redux, Vuex, or Context API	Redux, Vuex, React Context	Handling complex state in large applications	What are the benefits of using a state management library like Redux?

3. Backend Technologies

Skill	Description	Tools	Use Case	Interview Question
Database Management	Understanding relational and non-relational databases, schema design, querying, indexing, transactions, disk and memory data structures	PostgreSQL, MySQL, MongoDB, SQLite	Designing and optimizing database systems for applications	How do you ensure data consistency in a distributed database?
APIs and Protocols	Principles of RESTful APIs, RPC, GraphQL, API security, versioning, documentation, error handling	Swagger, GRPC. Postman, GraphQL	Building scalable and maintainable APIs for frontend-backend communication	What are the key considerations when designing a RESTful API?
Authentication & Authorization	Implementing secure user authentication (OAuth, JWT) and role-based access control	OAuth, JWT, Passport.js - replace	Securing applications and managing user permissions	Explain the difference between authentication and authorization
Server-Side Frameworks	Knowledge of frameworks like Express.js, Django, Spring Boot for building backend services	Express.js, Django, Spring Boot	Developing server-side logic and handling requests	How does middleware work in Express.js?
Distributed Systems	Concurrency, consistency, idempotency, exactly-once delivery, event loop, thread pools, asynchronous processing		Developing micro-services and distributed systems	How do you ensure idempotency in an API?

4. Cloud and DevOps

Skill	Description	Tools	Use Case	Interview Question
Cloud Services	S3, DynamoDB, Compute services, IAM basics	AWS, GCP, Azure	Hosting scalable apps	What's the difference between S3 and a traditional FS?
Containerization & Orchestration	Using Docker for containerizing applications and Kubernetes for orchestration	Docker, Kubernetes, Docker Compose	Managing microservices and ensuring consistent environments	Explain the benefits of using containers in software development
CI/CD Pipelines	Automating testing and deployment	GitHub Actions, GitLab CI	Deploying new features safely	What are the steps in a CI/CD pipeline?
Monitoring & Observability	Logs, metrics, tracing, alerting	Prometheus, Grafana, Sentry	Debugging production issues	How do you trace a request through a distributed system?
Infrastructure as Code	Managing cloud infrastructure using Terraform, CloudFormation, or ARM templates	Terraform, AWS CloudFormation, Azure Resource Manager	Automating infrastructure provisioning and management	What are the advantages of using Infrastructure as Code?

II. Blueprints for student projects

1. Project Blueprint: Classical ML

Step	Task	Description	Tools	Skills to Demonstrate	Example Scope
1	**Data Preparation**	Collect, clean, and format raw or unstructured data for modeling. This includes removing duplicates, handling missing values, encoding categorical variables, and feature scaling.	Pandas, NumPy	Data cleaning, preprocessing pipelines, dealing with missing or inconsistent data, data type conversions.	Prepare a real estate dataset (e.g., house prices) by cleaning and engineering features like square footage, location, and date.
2	**Model Selection & Hyperparameter Optimization (HPO)**	Choose appropriate regression models (e.g., linear regression, decision trees, or ensemble methods), and fine-tune hyperparameters to improve performance.	Scikit-learn	Model selection logic, cross-validation setup, GridSearchCV or RandomizedSearchCV for hyperparameter tuning.	Compare linear regression, random forest, and gradient boosting models on training data to predict housing prices.
3	**Model Training**	Train the selected model(s) on preprocessed training data. Monitor training curves if applicable and ensure correct data split practices.	Scikit-learn	Fit models, split data using train-test or train-validation-test, manage pipelines and transformers.	Train a regression model using 80/20 data split. Explore performance across training epochs if applicable.
4	**Evaluation**	Evaluate model performance using appropriate metrics (e.g., MAE, MSE, RÂ^2). Analyze errors, residuals, and assess whether the model generalizes well.	Scikit-learn metrics	Evaluation metrics, overfitting/underfitting analysis, residual error interpretation, visualization of actual vs. predicted.	Evaluate the best model on test set. Identify any signs of overfitting and visualize prediction errors.
5	**Visualization & Storytelling**	Present the results through plots and narrative. Communicate findings, highlight model performance, and explain decision impacts.	Matplotlib, Seaborn	Data and model visualization, storytelling, generating report-style charts (e.g., bar, scatter, residual plots).	Create charts comparing model predictions vs. actual prices. Present feature importances and model limitations.

2. Project Blueprint: Deep Learning

Step	Task	Description	Tools	Skills to Demonstrate	Example Scope
1	**Design & Implement Model**	Define the prediction problem, select input features, and design the architecture of a neural network (e.g., MLP). Implement the model and training loop.	PyTorch, Jupyter, Python	Designing neural network architectures, writing forward and training loops.	Build a neural network to predict house prices from tabular features (e.g., square footage, number of rooms).
2	**Data Preparation**	Load data from CSV, clean and normalize features, handle missing values, and split the dataset into training and validation sets.	Pandas, NumPy, Python	Data ingestion, scaling, encoding, stratified splits.	Prepare a structured dataset for training by normalizing numeric features and one-hot encoding categorical features.
3	**Train**	Train the neural network while monitoring training and validation loss. Tune hyperparameters like learning rate, number of hidden layers, and batch size.	PyTorch, Jupyter, Python	Training loop implementation, backpropagation, loss curve monitoring, tuning network depth/width.	Train an MLP on tabular customer data to predict churn risk.
4	**Evaluate**	Test the trained model on a hold-out test set. Use metrics like MAE, RMSE, or RÂ². Analyze prediction errors and diagnose issues like overfitting.	Pandas, NumPy, Python	Applying evaluation metrics, interpreting residuals, visualizing predictions vs. actuals.	Evaluate the trained model on unseen test data and identify patterns in mispredictions.
5	**Deploy & Monitor**	Package the model as a simple Streamlit app or REST API. Visualize predictions, enable input from users, and simulate monitoring.	Streamlit, Python	App development, model loading, basic UI design, handling user input.	Build a Streamlit app where users input tabular data and receive predictions with explanation.

3. Project Blueprint: Transformer

Step	Task	Description	Tools	Skills to Demonstrate	Example Scope
1	**Design & Implement Model**	Implement core components of a Transformer encoder block including Multi-Head Self-Attention, Feedforward layers, Layer Normalization, and Positional Encoding from scratch.	PyTorch	Understanding Transformer architecture, coding attention mechanisms, normalization layers, and custom modules.	Manually implement a minimal Transformer encoder block for character-level or word-level language modeling.
2	**Data Preparation**	Prepare a small text corpus for training. This includes text cleaning, tokenization, vocabulary creation, encoding, and batching, all of which are suitable for sequence tasks.	Python	Text preprocessing, building tokenizers, encoding sequences, batching for training.	Process a public domain text (e.g., Shakespeare or news snippets) into training-ready format.
3	**Train & Tune**	Train the implemented Transformer on a small dataset. Tune parameters such as learning rate, number of layers, and batch size. Monitor loss trends across epochs.	PyTorch	Writing training loops, loss calculation, optimizer setup, monitoring overfitting.	Train the model for next-word prediction on a tiny toy dataset.
4	**Evaluate**	Evaluate the trained model using loss curves and optionally perplexity. Visualize training vs. validation loss. Interpret learning behavior.	Matplotlib	Metric tracking, visualization, debugging over/underfitting, understanding convergence.	Plot training and validation loss, comment on training dynamics.
5	**Deploy & Monitor**	Wrap the trained model into a simple app using Streamlit. Allow users to input text and see next-token or text generation outputs. Simulate deployment.	Streamlit, Python	App/UI design, model inference integration, serving predictions.	Build a minimal web app that generates one word/token at a time given input text using the trained Transformer.

4. Project Blueprint: LLM API

Step	Task	Description	Tools	Skills to Demonstrate	Example Scope
1	**Problem Definition**	Identify a practical NLP challenge. Specify input/output behavior and success criteria.	NA	Problem formulation, requirement gathering, critical analysis	Define a product description generator based on bullet points
2	**System Design**	Select suitable frameworks (LangChain, RAG, agents). Design a modular architecture using prompt templates, memory, and context handling.	NA	System architecture, evaluating trade-offs, few-shot prompt engineering	Design a system to summarize documents using retrieval-augmented generation
3	**Implementation**	Code the pipeline using prompt tools and APIs. Include data parsing, chaining modules, and logic flow.	Python, LangChain, OpenAI API, Pandas, NumPy, Jupyter	API integration, chaining prompts, control flow, prompt debugging	Implement a Q&A assistant over a PDF or a company knowledge base
4	**Evaluation & Iteration**	Measure system quality using standard or custom metrics. Iterate by adjusting prompts or models to improve results.	RAGAS, Python	Evaluation metrics, hallucination detection, prompt refinement, model comparison	Use automated metrics and human judgment to improve chatbot answers
5	**Deployment & Storytelling**	Build a simple front-end to demo the tool. Highlight user input/output, purpose, and quality indicators.	Streamlit, Python	UI/UX design, data storytelling, presenting AI pipelines	Build a Streamlit app where users type a question and see generated answers with context

5. Project Blueprint: Agents

Step	Task	Description	Tools	Skills to Demonstrate	Example Scope
1	**Design and Implement Agent**	Define your agent's reasoning process, create prompt templates, and integrate tools. Develop chain-of-thought workflows and decision logic.	LangChain, LangSmith	Reasoning design, system integration, chain-of-thought prompting	Build an agent that summarizes and reasons over user manuals
2	**Integrate RAG**	Enable retrieval-augmented generation by linking your agent to a vector database or search API. Improve factual grounding.	Pinecone, Weaviate, Chroma, FAISS	Vector DB integration, prompt retrieval, query transformation	Connect to Pinecone and retrieve chunks from documentation for question answering
3	**Test and Evaluate**	Assess the accuracy and robustness of agent outputs. Inspect reasoning steps and validate end-to-end logic.	LangSmith, pytest	Agent evaluation, trace inspection, bug diagnosis	Write tests for evaluating agent outputs with edge cases and hallucination checks
4	**Implement Guardrails and Safety**	Add runtime safety features such as prompt filters, input sanitization, and content moderation to avoid harmful or biased outputs.	Guardrails.ai, Rebuff	Prompt safety, filtering policies, risk mitigation	Add rejection conditions for inappropriate or unsafe inputs and outputs
5	**Deploy and Monitor**	Package the final agent, add observability tools (e.g., logging, latency tracking), and deploy it as a service or application.	LangSmith, FastAPI, Streamlit	Deployment, monitoring, service design	Deploy the agent on a web app with logs, feedback capture, and retry logic

6. Project Blueprint: Computer Vision

Step	Task	Description	Tools	Skills to Demonstrate	Example Scope
1	**Data Preparation**	Collect, clean, and resize images; split them into train/validation folders; optionally create text prompts or captions.	Python, OpenCV, Pillow	Image I/O, resizing, augmentation, dataset organization	Load a custom image set, resize, save in a structured directory for model training.
2	**Model Selection & Hyperparameter Tuning**	Select an appropriate diffusion-based vision model (e.g., Stable Diffusion, DreamBooth) and design a hyperparameter search strategy (including learning rate, batch size, and scheduler).	HuggingFace diffusers, Optuna	Understanding vision diffusion architectures, configuring training arguments, running HPO loops	Compare SD v1.5 and SDXL checkpoints; run a grid search over learning rates.
3	**Training / Fine-Tuning**	Finetune the chosen diffusion model on the prepared dataset; monitor loss and sample intermediate generations.	diffusers pipelines, PyTorch Lightning, Accelerate	Writing training scripts, managing GPUs, checkpointing, sampling latents to images	Fine-tune Stable Diffusion so it generates branded product photos that match company style.
4	**Evaluation**	Assess model quality using quantitative metrics (e.g., FID, CLIP score) and qualitative visual inspection; analyze prompt fidelity.	Matplotlib, CLIP, Seaborn	Computing vision metrics, plotting distributions, critiquing generated images	Plot FID vs. training epochs and review a gallery of generated images for accuracy to prompts.
5	**Visualization & Demo**	Build an interactive demo to showcase the model; allow users to enter prompts and view generated images.	Streamlit, Python	Web UI creation, model inference API, storytelling with visuals	Deploy a Streamlit app where users enter a prompt and see four generated images side-by-side.

III. Essential Soft Skills

Soft Skills

#	Skill	Extended Description	Example in Student Project
1	**Justify technical choices**	Explain and defend why a particular model, tool, or architecture is chosen over alternatives based on performance, scalability, or interpretability.	Student compares Random Forest vs XGBoost and selects XGBoost due to better validation performance on imbalanced dataset.
2	**Critical thinking**	Identify flaws or inefficiencies in code, data, or logic and propose a reasoned fix or redesign.	Student finds that the model overfits due to data leakage and fixes the pipeline to prevent target leakage.
3	**Planning project**	Create a timeline, define work packages, estimate effort, and monitor progress using task tracking tools or spreadsheets.	Student outlines a Gantt chart with stages for data prep, modeling, evaluation, and app deployment.
4	**Data-based storytelling**	Use visuals (charts, dashboards) to present insights, trends, or outcomes in a compelling narrative.	Student builds a Streamlit dashboard showing model predictions and confidence intervals for user inputs.
5	**Framing app to tech task**	Translate high-level user needs into specific machine learning problems and measurable objectives.	Student turns the need for 'auto-tagging customer complaints' into a multi-label classification task.
6	**Obtain stakeholder buy-in**	Prepare proposals and communicate value clearly to team members, instructors, or external reviewers.	Student pitches a project idea to peers, articulating goals, benefits, and feasibility.
7	**Business acumen**	Understand how project decisions impact costs, resources, or business KPIs.	Student selects a simpler model with slightly lower accuracy but 5x faster inference for low-latency deployment.
8	**Product management**	Write product requirement documents, define user stories and iterate based on feedback.	Student defines a user story for adding a file-upload feature to test predictions on unseen data.
9	**Innovation / Creativity**	Propose novel ideas or tackle open-ended problems without a known solution path.	Student experiments with adding synthetic data to improve robustness of model to rare events.
10	**Literature / Technology research**	Identify and summarize relevant academic papers, blog posts, or repositories to guide implementation.	Student reviews recent Hugging Face blog posts to understand prompt tuning for sentiment classification.