NEXIQUILL

WHERE IQ MATTERS

NEXIQUILL ACADEMY / COURSE BROCHURE

Python Full Stack with Generative AI

A practical, instructor-led 12-week curriculum. Learn full-stack development, build real applications, and create a portfolio-ready project with AI features.

12 WEEKS
Live instructor-led classes

REAL PROJECTS
Build and deploy a portfolio app

CAREER SUPPORT
Interview preparation included

FIRST-BATCH OFFER / FIRST 50 SEATS

INR 14,999

Enrollment closes 18 November 2026

REGULAR FEE

~~INR 33,999~~

CLEAR PAYMENT BREAKDOWN

Upfront seat-booking fee	**INR 299**
Remaining after receiving an offer letter	**INR 14,700**

The remaining balance is due after receiving an offer letter.

PAID INTERNSHIP OPPORTUNITY

A 3-month paid internship opportunity may follow the program, with an advertised stipend up to INR 15,000 per month. Internship selection and stipend depend on the offer.

WhatsApp +91 95053 25418

Internship and employment outcomes are conditional, not guaranteed. Confirm eligibility, payment schedule, refund terms, and other commercial terms in writing before enrolling.

NEXIQUILL

PYTHON FULLSTACK WITH GENERATIVE AI

12-Week Candidate Curriculum & Delivery Plan

Learn the foundations. Build real applications. Deploy a portfolio project.

PROGRAM AT A GLANCE

Duration	12 weeks (3 months)	**Delivery**	Live instructor-led classes + guided labs
Core focus	Python, Django/DRF, React, Node.js basics, databases, GenAI and deployment	**Learning style**	Concepts → coding demos → practice → reviews
Outcome	Portfolio-ready full-stack application with AI feature	**Evidence**	GitHub repository, documentation and demo

What candidates can expect

- A structured progression from programming fundamentals to full-stack development and practical Generative AI integration.
- Hands-on exercises every week, code reviews, checkpoints and a final project presentation.
- Guidance on Git/GitHub, debugging, API integration, deployment, documentation and interview preparation.
- A realistic learning workload: attend live sessions, practise independently, complete assignments and ask questions early.

Important: This document describes the planned learning pathway. Specific class timings, instructor allocation, assessment dates, internship selection and commercial terms should be confirmed by NexiQuill in writing. Internship or employment outcomes are not guaranteed.

1. How the Classes Will Run

Suggested weekly rhythm (the institute may publish the final timetable separately):

Session type	What happens	Typical time
Concept class	Instructor explains the topic, diagrams, examples and common mistakes.	60–90 min
Live coding	Instructor builds a small feature and explains decisions step by step.	30–60 min
Guided lab	Candidates implement the same idea, then extend it independently.	45–60 min
Review / doubt clearing	Review code, discuss blockers, test solutions and set next tasks.	Weekly

Candidate responsibilities

- Bring a laptop, stable internet and a working development environment; install tools before class where possible.
- Attend sessions, take notes, submit weekly assignments and keep code in Git/GitHub.
- Ask for help when blocked; explain what was attempted and share the relevant error or code.
- Use AI tools responsibly: verify generated code, understand it, test it and disclose AI assistance where required.

2. Technology Stack & Where It Is Used

Area	Technologies / tools	Purpose
Programming	Python, JavaScript, TypeScript basics	Core logic, scripting, browser interactions and typed frontend code.
Frontend	HTML5, CSS3, responsive design, React.js, React Router, fetch/axios	Build interactive pages, forms, reusable components and API-connected UI.
Backend	Django, Django REST Framework; Node.js/Express overview	Build REST APIs, validation, business logic, authentication and integrations.
Data	SQL, PostgreSQL, MySQL; MongoDB overview	Model, query, validate and persist application data.
GenAI	LLM APIs, prompt design, embeddings/RAG concepts, vector database concepts	Add AI-assisted search, Q&A; or summarisation to an application.
Engineering	Git, GitHub, Postman, testing, environment variables, debugging	Collaborate, inspect APIs, test features and manage changes safely.
Deployment	Cloud hosting concepts, frontend/backend deployment, logs and README	Publish a demo, configure environments and explain the architecture.

WEEK 1 | Programming Foundations, Setup & Git

Weekly goal: Set up a reliable development environment and learn to write, run and debug basic Python programs.

Topics & concepts

- Program structure, variables, data types, operators, input/output and type conversion.
- Conditions, loops, functions, scope, strings, lists, tuples, sets and dictionaries.
- Errors, exceptions, debugging approach and writing readable code.
- Install Python, VS Code, browser tools and Git; terminal basics and virtual environments.
- Git workflow: init/clone, status, add, commit, branch, push and README basics.

Hands-on lab: Create a command-line expense tracker or student marks analyser using functions, collections and input validation.

Expected deliverable: GitHub repository with working Python program, README, sample runs and at least 10 small practice problems.

Weekly checkpoint: Short fundamentals quiz + code walkthrough.

WEEK 2 | Core Python, OOP & Problem Solving

Weekly goal: Become comfortable organising Python code and solving common programming tasks.

Topics & concepts

- Modules, packages, imports, pip, virtual environments and dependency files.
- File handling, JSON/CSV basics, exceptions and logging introduction.
- Object-oriented programming: classes, objects, constructors, inheritance, composition and encapsulation.
- List/dictionary comprehensions, iterators/generators introduction and useful standard-library modules.
- Problem-solving patterns: searching, sorting, counting, string processing and time-complexity intuition.

Hands-on lab: Build a file-based contact or inventory manager with add, update, search, delete and export features.

Expected deliverable: Modular Python application with error handling, sample data and clear setup instructions.

Weekly checkpoint: Practical coding exercise + OOP explanation.

WEEK 3 | Web Foundations, HTML/CSS & JavaScript

Weekly goal: Understand how a web page works before building React interfaces.

Topics & concepts

- Client/server basics, HTTP request/response, browser developer tools and semantic HTML.
- Forms, accessibility basics, CSS selectors, box model, Flexbox, Grid and responsive layouts.
- JavaScript variables, functions, arrays, objects, conditions, loops and modules.
- DOM events, form validation, promises, async/await and fetch API.
- JSON, status codes, browser storage and handling loading/error states.

Hands-on lab: Create a responsive course registration page with client-side validation and a mock API/data flow.

Expected deliverable: Mobile-friendly static web project with reusable styles and Git history.

Weekly checkpoint: Responsive UI review + JavaScript fundamentals quiz.

WEEK 4 | React.js & Frontend Application Development

Weekly goal: Build component-based user interfaces and connect them to API data.

Topics & concepts

- Node/npm basics, Vite project setup, JSX, components, props and state.
- Events, conditional rendering, lists/keys, forms and controlled inputs.
- useState, useEffect, component lifecycle concepts and custom hooks introduction.
- React Router, layout patterns, reusable components and basic state organisation.
- API calls, loading/error/empty states, environment variables and frontend debugging.
- Styling strategy: responsive CSS; optional utility CSS concepts where appropriate.

Hands-on lab: Build a dashboard with navigation, searchable/filterable records, forms and API/mock-data integration.

Expected deliverable: React app with multiple routes, reusable components, responsive layout and deployment-ready configuration.

Weekly checkpoint: Feature demonstration + code review.

WEEK 5 | SQL, Relational Databases & Data Modelling

Weekly goal: Design a sensible data model and query it safely.

Topics & concepts

- Tables, rows, columns, primary/foreign keys, constraints and relationships.
- SELECT, WHERE, ORDER BY, GROUP BY, HAVING, joins and aggregate functions.
- INSERT, UPDATE, DELETE, transactions and basic indexing concepts.
- Schema design, normalisation intuition, ER diagrams and handling timestamps.
- PostgreSQL or MySQL setup; connect application requirements to data models.
- MongoDB document model overview and when relational vs document databases may fit.

Hands-on lab: Design a database for a learning platform: candidates, courses, enrolments, assignments and submissions.

Expected deliverable: ER diagram, schema/SQL script, sample dataset and a query collection.

Weekly checkpoint: SQL practical covering joins, grouping and relationships.

WEEK 6 | Django & REST API Development

Weekly goal: Create a structured Python backend that exposes data through REST endpoints.

Topics & concepts

- Django project/app structure, settings, URLs, views, models and migrations.
- ORM basics, relationships, querysets, admin and validation.
- Django REST Framework serializers, API views/viewsets and routers.
- REST principles, HTTP methods/status codes, pagination, filtering and error responses.
- Environment configuration, secrets, CORS concepts and Postman API testing.
- Authentication/permissions overview and secure handling of input.

Hands-on lab: Build course and assignment APIs with create/read/update/delete operations and validation.

Expected deliverable: Django REST API, database migrations, Postman collection and endpoint documentation.

Weekly checkpoint: API demo + endpoint testing.

WEEK 7 | Full-Stack Integration, Authentication & Testing

Weekly goal: Connect the React frontend to the Django API and create a coherent end-to-end workflow.

Topics & concepts

- Connect frontend API service functions to backend endpoints.
- Loading, success, empty and error states; form validation and user-friendly feedback.
- Authentication concepts, login/logout flow, permissions and protected routes.
- Testing fundamentals: unit vs integration tests; backend API tests and frontend component checks.
- Debugging across browser, API client and backend logs; handling CORS and configuration issues.
- Code organisation, reusable API helpers, naming, linting and review etiquette.

Hands-on lab: Integrate login and course enrolment flows (using an appropriate training-safe authentication setup).

Expected deliverable: Working full-stack application with documented API contract and basic tests.

Weekly checkpoint: End-to-end walkthrough + bug-fix exercise.

WEEK 8 | Node.js, Express & Alternative Backend Patterns

Weekly goal: Understand the Node.js ecosystem and how it compares with the Python/Django approach.

Topics & concepts

- JavaScript runtime, npm packages, modules and asynchronous programming review.
- Express app structure, middleware, routes, controllers and request validation.
- Build simple REST endpoints and connect to a database or mock repository.
- Authentication/security overview, error middleware and environment configuration.
- Compare Django/DRF and Express: conventions, strengths and suitable use cases.
- Introduction to backend service boundaries and when not to duplicate services.

Hands-on lab: Create a small Express API for a task tracker, then document equivalent Django concepts.

Expected deliverable: Working mini API, request collection and a short technology comparison note.

Weekly checkpoint: Explain request flow, middleware and API design.

WEEK 9 | Generative AI Foundations & LLM API Integration

Weekly goal: Understand practical LLM integration and add a bounded AI feature to an application.

Topics & concepts

- What LLMs can/cannot do; tokens, context windows, latency, cost and model selection concepts.
- Prompt design: clear instructions, context, output formats, constraints and examples.
- Call an LLM API from the backend; protect API keys using environment variables.
- Structured outputs, response validation, timeouts, retries and graceful failure.
- Privacy, data minimisation, bias, hallucinations, human review and responsible use.
- Build an AI feature such as course FAQ answers, summarisation or guided content generation.

Hands-on lab: Add an AI-powered study assistant endpoint with input limits, validation and user-facing error states.

Expected deliverable: Working AI integration, configuration guide, test prompts and documented limitations.

Weekly checkpoint: Feature demo + responsible-AI review.

WEEK 10 | Embeddings, RAG & AI-Powered Search

Weekly goal: Learn the basic retrieval workflow for grounding AI answers in a controlled knowledge source.

Topics & concepts

- Embeddings and semantic similarity; difference between keyword and semantic search.
- Document preparation, chunking, metadata and retrieval quality basics.
- RAG flow: ingest documents → embed → retrieve relevant chunks → construct context → generate answer.
- Vector database concepts, including Qdrant or an equivalent learning environment (subject to setup).
- Grounding responses in retrieved content, citations/source snippets and handling no-answer cases.
- Evaluation basics: representative questions, relevance, correctness and failure cases.

Hands-on lab: Create a small knowledge-base Q&A; prototype using provided course/FAQ documents.

Expected deliverable: RAG prototype or equivalent retrieval demo, architecture diagram and evaluation examples.

Weekly checkpoint: Candidate explains each RAG stage and demonstrates a grounded answer.

WEEK 11 | Capstone Build, Deployment & Quality

Weekly goal: Turn the accumulated skills into a demonstrable, documented full-stack project.

Topics & concepts

- Choose a scoped capstone and write user stories, acceptance criteria and task breakdown.
- Implement core UI, backend endpoints, database schema and validation.
- Integrate one clearly scoped GenAI feature; keep normal app functionality usable if AI fails.
- Testing, bug triage, security review, responsive checks and edge-case handling.
- Deploy frontend and backend using suitable hosting options; configure environment variables.
- Logs, basic monitoring concepts, seed/demo data, README, setup instructions and known limitations.

Hands-on lab: Build and deploy the capstone in small reviewed milestones; conduct a peer/instructor code review.

Expected deliverable: Public or reviewer-accessible demo, source repository, README, API documentation and test evidence.

Weekly checkpoint: Milestone review: functionality, code quality, usability and deployment.

WEEK 12 | Capstone Completion, Portfolio & Interview Preparation

Weekly goal: Complete the project, communicate technical decisions and prepare to demonstrate skills.

Topics & concepts

- Fix priority bugs, polish UI, improve empty/error/loading states and finalise the data model.
- Final integration testing, deployment verification and configuration/security checklist.
- Write a project README: problem, features, architecture, stack, setup, screenshots and limitations.
- Prepare a 5–10 minute project demo and explain key technical decisions.
- Interview practice: Python, JavaScript/React, APIs, SQL, debugging, Git and basic GenAI/RAG concepts.
- Resume/GitHub review, mock interview, feedback and next-step learning plan.

Hands-on lab: Final project presentation, live demo, code walkthrough and question-and-answer session.

Expected deliverable: Completed capstone, polished GitHub repository, deployment link where feasible, project presentation and personal improvement plan.

Weekly checkpoint: Final rubric assessment; completion/certificate eligibility follows institute policy.

3. Capstone Project Options

Candidates can choose one project after discussing scope with the instructor. The project should demonstrate frontend, backend, database and a practical AI or retrieval feature without relying on AI for the entire application.

AI-Powered Course Assistant

Candidate login, course/FAQ knowledge base, semantic retrieval, grounded Q&A;, feedback and admin document management.

Smart Support Ticket Portal

Ticket creation, status workflow, searchable history, dashboard, AI ticket summary/category suggestion and role-based access.

Learning & Assignment Tracker

Course catalogue, enrolments, assignments, submissions, progress dashboard and AI-assisted study summaries.

Knowledge-Base Search App

Upload or manage approved documents, chunk/index content, retrieve relevant passages and answer with source references.

Minimum capstone acceptance checklist

- Clear problem statement and a manageable list of user stories.
- Responsive React UI with navigation, forms and loading/error/empty states.
- Backend REST API with validation and consistent responses.
- Database schema with meaningful relationships and sample data.
- At least one tested end-to-end workflow.
- One bounded GenAI/RAG feature with limitations and failure handling documented.
- Git history, README, setup instructions, environment-variable template and demo walkthrough.
- No secrets, private API keys or sensitive real user data committed to the repository.

4. Assessment & Completion Framework

Assessment	Suggested weight	What is reviewed
Weekly labs & assignments	25%	Correctness, completion, code clarity and consistency.
Module checkpoints	20%	Python, web, SQL, API, React and integration fundamentals.
Code quality & Git workflow	15%	Readable structure, commits, README, debugging and testing habits.
Capstone project	30%	Working features, integration, AI feature, deployment and documentation.
Demo & technical explanation	10%	Ability to explain decisions, trade-offs, limitations and next steps.

These percentages are a suggested framework, not a published grading policy. The institute should confirm the official passing threshold, attendance requirements, resubmission rules and certificate criteria before distribution.

5. Candidate Outcomes by the End of 12 Weeks

- Write and debug Python programs using functions, collections, modules and OOP fundamentals.
- Build responsive React interfaces and integrate them with REST APIs.
- Design relational database tables and write common SQL queries.
- Create Django REST endpoints and understand the basic Express/Node.js request flow.

- Use Git/GitHub, API testing tools and a repeatable development workflow.
- Integrate an LLM API and explain the fundamentals of embeddings and RAG.
- Deliver a documented full-stack project and present it in a technical walkthrough.

6. Internship, Stipend & Career Notes

The promotional poster mentions a three-month paid internship opportunity, remote work, a stipend of up to ■15,000 per month, and possible consideration of top performers for full-time roles in the ■6.9–11 LPA range, subject to performance, availability and selection. These are conditional opportunities—not guaranteed outcomes for every candidate.

Before enrolment, candidates should request written confirmation of eligibility criteria, selection process, internship start date and duration, work expectations, stipend amount and payment schedule, evaluation criteria, any agreement/conditions, and whether the internship is separate from the training program.

The poster also shows a first-batch fee of INR 14,999, an enrollment fee of INR 299, and INR 14,700 described as payable after receiving an offer letter. Candidates should confirm the exact fee schedule, refund/cancellation policy, taxes if applicable, and what happens if an offer letter is not made. Do not treat a poster as a substitute for the institute's written terms.

7. Suggested Weekly Candidate Routine

Day / cadence	Suggested activity
Class days	Attend the live lesson, follow the coding demo, take notes and reproduce the key example.
Practice after class	Spend 45–90 minutes implementing exercises without copying the full solution.
Weekly lab day	Complete the mini-project, push changes to GitHub and write down blockers.
Weekly review	Demo the work, explain code choices, review feedback and plan improvements.
Ongoing	Keep a learning log, maintain a clean repository and revisit weak concepts.

Build skills. Create opportunities. Show what you can build.

NexiQuill • Candidate Curriculum • Version 1.0 • Prepared as a proposed 12-week delivery plan based on the course information provided.