

KAUR'S CODING ACADEMY

Complete Coding Roadmap and Syllabus

For Grades 3-12, College Students, and Working Professionals

Category	Details
Mode of Delivery	Online live classes with practical coding, guided projects, homework tasks, and project presentations
Target Regions	USA, Australia, Canada, Singapore, and India
Core Outcomes	Real-world websites, practical apps, interactive games, portfolio projects, deployment, and interview readiness
Learning Approach	Simple explanation, live demonstration, hands-on practice, weekly revision, and age-wise project building

Academy Promise

Every learner should understand the concept, write the code with confidence, and finish each stage with a project they can proudly show to parents, schools, colleges, clients, or employers.

Prepared as a professional curriculum framework for online coding programs

Contents

1. Academy Overview and Learning Vision
2. Target Audience and Online Market Positioning
3. Complete Learning Roadmap
4. Age-Wise Curriculum for Grades 3-12
5. College and Professional Career Tracks
6. Detailed Syllabus by Technology
7. Real-World Website, App, and Game Project Roadmap
8. Teaching Methodology for Better Understanding
9. Assessment, Certificates, and Portfolio System
10. Recommended Class Format and Delivery Plan

1. Academy Overview and Learning Vision

Kaur's Coding Academy is designed as a practical online coding academy where students do not only learn syntax, but also learn how to think, plan, build, test, improve, and present real digital products. The curriculum begins with simple logic and creativity for younger learners and gradually moves toward professional web development, mobile app development, backend systems, databases, deployment, and interview preparation.

The complete learning path has been divided into clear levels so that a Grade 3 learner, a high school student, a college student, and a working professional can all join at the right stage. Each level includes concepts, practice tasks, mini-projects, revision, and one portfolio project.

Core Learning Outcomes

- Build confidence with computers, typing, problem solving, and coding logic.
- Create real-world websites using HTML, CSS, JavaScript, and responsive design.
- Build interactive apps using JavaScript, Python, React, React Native, and backend APIs.
- Create games that teach loops, conditions, events, scoring, movement, collision, and debugging.
- Understand databases, APIs, authentication, deployment, version control, and teamwork.
- Finish every major level with a project that can be added to a portfolio.

Curriculum Pillars

Pillar	What It Means in Class	Student Output
Logic First	Students learn patterns, sequencing, decisions, loops, and problem breakdown before heavy syntax.	Flowcharts, puzzles, mini challenges, simple games
Build While Learning	Every important concept is connected with a visible output so students understand why they are learning it.	Web pages, animations, calculators, quizzes, apps
Project-Based Practice	Each module ends with a small or medium project, followed by review and improvements.	Portfolio website, API app, game, mobile app, backend project
Professional Habits	Older learners use Git, GitHub, folder structure, debugging, documentation, and deployment.	Hosted projects, GitHub repository, final demo
Communication	Students explain what they built, how it works, and what they improved after feedback.	Project presentation and viva-style discussion

2. Target Audience and Online Market Positioning

The academy can run age-wise batches as well as skill-wise batches. This avoids mixing very young learners with advanced learners and helps every student learn at a comfortable pace.

Audience	Primary Need	Best Program Positioning
Grades 3-5	Fun, creativity, computer confidence, logic, and early coding habits.	Junior Coding Explorer - creative coding, games, animations, simple web pages
Grades 6-8	Stronger logic, web basics, interactive projects, and game development.	Creative Coding Builder - HTML, CSS, JavaScript, Python basics, websites and games
Grades 9-10	Real programming, web development, school projects, and portfolio building.	Teen Web and App Developer - JavaScript, Python, React basics, APIs, GitHub
Grades 11-12	Career clarity, college readiness, internships, and serious project work.	Career Starter Developer - full-stack foundations, deployment, capstone projects

Audience	Primary Need	Best Program Positioning
College Students	Job-ready skills, internships, portfolio, and interview preparation.	Full-Stack, Mobile App, Python/Django, or Java Backend tracks
Professionals	Upskilling, switching domains, freelancing, or building products.	Fast-track practical bootcamps with real projects, deployment, and interviews

Target Areas

Online classes can be positioned for USA, Australia, Canada, Singapore, and India with flexible batch timings, weekend cohorts, after-school classes, and project review sessions.

- For school students: after-school and weekend coding batches.
- For college students: evening/weekend skill-building and internship-ready cohorts.
- For professionals: fast-track live training with project reviews and career support.

3. Complete Learning Roadmap

The roadmap below shows how a learner can move from beginner-level thinking to real-world product development. Students do not need to complete every track; they can choose the branch that matches their goal after completing the foundation levels.

Level	Stage	Topics	Best Fit
Level 1	Digital Confidence and Logic	Computer basics, typing habits, files/folders, internet safety, sequences, patterns, debugging mindset	Grades 3-5 beginners
Level 2	Creative Coding and Games	Block coding, events, sprites, loops, score, timers, conditions, animations, simple game mechanics	Grades 3-6
Level 3	Web Design Foundations	HTML structure, text, links, images, lists, tables, forms, CSS selectors, colors, spacing, box model	Grades 5-8
Level 4	Responsive Websites	Flexbox, Grid, responsive layout, transitions, animations, landing pages, portfolio websites	Grades 6-9
Level 5	JavaScript Programming	Variables, operators, arrays, objects, functions, conditions, loops, DOM, events, forms, local storage	Grades 7-10
Level 6	Python Programming	Variables, data types, strings, inputs, functions, conditions, loops, lists, tuples, file handling, OOP basics	Grades 7-12
Level 7	Frontend Development	React, components, props, state, hooks, forms, routing, API calls, optimization, deployment	Grades 9-12, college
Level 8	Backend and Databases	Node.js/Express or Django/Java, REST APIs, MongoDB/SQL, authentication, Postman, MVC, security basics	Grades 10-12, college, professionals
Level 9	Mobile App Development	React Native, Expo, navigation, mobile UI, device features, storage, API integration, app deployment	College, professionals, advanced teens
Level 10	Capstone and Career	Full-stack integration, GitHub, deployment, documentation, presentation, interview preparation	College, professionals, advanced teens

Roadmap Note

For younger children, the goal is not to rush into advanced frameworks. The goal is to build confidence, logic, creativity, and consistency. Advanced tools such as React, Node.js, Django, Java, and React Native are introduced only when students are ready for structured project development.

4. Age-Wise Curriculum for Grades 3-12

Grades 3-5: Junior Coding Explorer

This track is built for young learners who need friendly explanations, visual learning, and quick creative outputs. Classes should feel like building, not memorizing.

Module	Topics to Teach	Class Activity	Project Output
1. Computer Confidence	Keyboard, mouse/trackpad, files, folders, browser use, safe search, online class discipline	Name folders, save images, open links, submit homework screenshots	My Digital Notebook
2. Logic Without Coding	Sequences, patterns, step-by-step instructions, cause and effect, mistakes and debugging	Human robot game, flowchart stories, logic puzzles	Algorithm poster
3. Block Coding Basics	Sprites, events, motion, looks, sounds, loops, conditions, score, timer	Build a moving character and interactive story	Animated story
4. Game Thinking	Game goal, player, enemy, score, lives, levels, collisions, restart button	Guided game building with teacher support	Maze or Catch Game
5. Beginner Web Pages	HTML headings, paragraphs, images, links, colors, simple styling	Create a page about favorite hobby, sport, or animal	My First Web Page
6. Python Fun	Print, variables, input, numbers, strings, turtle drawing, simple loops	Draw shapes, ask questions, print results	Python Art Card
7. Mini App Logic	Input-output, if/else decisions, simple quiz, score counting	Make a quiz about school, animals, or space	Quiz Game
8. Final Showcase	Revision, polishing, presentation skills, explaining code simply	Practice demo and answer simple questions	Parent Demo Project

Grades 6-8: Creative Coding Builder

This track adds real coding syntax while keeping projects visual and interesting. Students learn how websites, small apps, and games work in the browser.

Phase	Duration	Topics to Teach	Projects
Phase 1: Web Basics	6-8 weeks	HTML structure, text formatting, links, images, lists, tables, forms, CSS colors, fonts, margin, padding, box model	Personal profile page, school club website
Phase 2: Modern Layouts	4-6 weeks	Flexbox, Grid, responsive design, transitions, animations, media queries	Responsive landing page, event website
Phase 3: JavaScript Basics	8-10 weeks	Variables, data types, operators, arrays, objects, functions, conditions, loops	Calculator, marks grade checker, random quote generator

Phase	Duration	Topics to Teach	Projects
Phase 4: DOM and Events	6-8 weeks	DOM selection, changing text/styles, event listeners, keyboard/mouse events, form validation, local storage	To-do list, quiz app, memory card game
Phase 5: Python Foundations	6-8 weeks	Variables, inputs, strings, numbers, conditions, loops, lists, functions, file basics	Number guessing game, word counter, mini report generator
Phase 6: Project Sprint	4 weeks	Planning, wireframe, coding, testing, debugging, presentation	Website + interactive app or game

Grades 9-10: Teen Web and App Developer

This track is suitable for students who can handle structured coding and want serious project outcomes. It introduces APIs, GitHub, React basics, and practical project planning.

Month	Learning Focus	Topics to Teach	Portfolio Output
1	HTML, CSS, and Responsive Design	Semantic HTML, forms, CSS selectors, Flexbox, Grid, media queries, UI sections, accessibility basics	Responsive portfolio website
2	JavaScript Core	Variables, arrays, objects, functions, conditions, loops, debugging, problem solving	Logic challenge set
3	DOM, Events, and Browser Apps	DOM manipulation, events, validation, local storage, timers, date/time, error handling	Quiz app or expense tracker
4	APIs and GitHub	Fetch/Axios basics, JSON, async/await, API data display, Git, GitHub, README writing	Weather or movie search app
5	React Basics	Components, JSX, props, state, events, conditional rendering, lists, forms	React product catalog
6	React Projects	Hooks, useEffect, routing, reusable components, API integration, deployment	Multi-page React app
7	Backend Introduction	Client-server model, HTTP methods, APIs, Node.js or Django introduction	Simple API connected with frontend
8	Capstone and Demo	Planning, coding, testing, deployment, presentation	Full project demo with GitHub link

Grades 11-12: Career Starter Developer

This track prepares senior students for college, internships, freelancing basics, and career-oriented project work.

Quarter	Focus	Topics to Teach	Major Output
Q1	Strong Web Development	Advanced HTML/CSS, responsive UI, JavaScript ES6, DOM, APIs, Git/GitHub	Professional portfolio + API app
Q2	Frontend Framework	React components, props, state, hooks, forms, routing, Context API, API integration, deployment	React dashboard or e-commerce frontend

Quarter	Focus	Topics to Teach	Major Output
Q3	Backend and Database	Node.js/Express or Django, REST APIs, MongoDB/SQL, CRUD, MVC, authentication, authorization, Postman	Task manager or user system backend
Q4	Full-Stack Capstone	Frontend-backend integration, environment variables, hosting, testing, documentation, project presentation	Full-stack website/app with live demo

5. College and Professional Career Tracks

College students and working professionals usually need focused outcomes: job-ready skills, strong projects, deployment, GitHub profile, interview preparation, and confidence to build real products. The following tracks can run independently or as bundled programs.

Track	Recommended Duration	Technologies	Best For	Capstone Project
Full-Stack Web Development	7-9 months	HTML, CSS, JavaScript, React, Node.js, Express, MongoDB/SQL, Git, deployment	College students, job seekers, freelancers	E-commerce platform, LMS, job portal, booking system
Mobile App Development	8 months	HTML/CSS/JS foundation, React Native, Expo, Node.js, MongoDB, APIs, deployment	Mobile app learners, freelancers, startups	Fitness tracker, news app, food order app, task app
Python and Django Backend	5-7 months	Python, OOP, file handling, Django, ORM, forms, APIs, authentication, SQL	Backend learners, college students, automation beginners	User dashboard, blog platform, CRM, API backend
Java Backend Developer	8-10 months	Core Java, OOP, collections, JDBC, JSP/Servlet, Spring, Spring Boot, Hibernate, microservices basics	Engineering students and backend professionals	Spring Boot REST API with database and authentication
Professional Fast-Track	12-20 weeks	Customized path based on current skill level: React, Node.js, Python/Django, Java/Spring, or React Native	Working professionals and career switchers	One production-style project with GitHub and deployment

Suggested Entry Points

- Absolute beginners should start with HTML, CSS, JavaScript, and basic programming logic.
- Students with basic web knowledge can start with React and then move to backend development.
- Learners who prefer backend can choose Python/Django or Java/Spring after programming fundamentals.
- Mobile app learners should learn JavaScript and React basics before React Native.
- Professionals should begin with a diagnostic assignment and then join a focused fast-track plan.

6. Detailed Syllabus by Technology

This syllabus bank can be used to create separate courses or combined learning paths. Topics are arranged from beginner to advanced so teachers can adjust speed based on age and batch level.

6.1 Web Design: HTML and CSS

Module	Topics to Teach	Practice and Project
HTML Basics	HTML document structure, doctype, meta tags, headings, paragraphs, spans, comments, attributes	Create a personal introduction page
Text, Links, and Media	Text formatting, anchor tags, images, alt text, width/height, audio/video basics	Build a hobby or travel page
Lists, Tables, and Forms	Ordered/unordered lists, nested lists, table rows/cells/headers, forms, inputs, textarea, select, checkboxes, radio buttons	Contact form and timetable page
CSS Basics	Inline, internal, external CSS, selectors, class, id, colors, fonts, backgrounds, margins, padding	Style the personal website
Box Model and Positioning	Content, padding, border, margin, display, static/relative/absolute positioning	Create cards and sections
Modern Layouts	Flexbox, Grid, fixed/fluid/responsive layouts, spacing systems	Responsive landing page
Effects and Responsive Design	Transitions, hover effects, keyframe animations, media queries, mobile-first design	Animated service section
Final Project	Planning, wireframe, coding, testing, review, presentation	Portfolio website or business landing page

6.2 JavaScript Programming

Module	Topics to Teach	Practice and Project
JavaScript Setup	Purpose of JavaScript, inline/internal/external JS, linking files, console, comments	First interactive button
Variables and Data Types	var, let, const, strings, numbers, booleans, null, undefined, type conversion	Student profile data program
Operators and Expressions	Arithmetic, comparison, logical operators, template literals	Bill calculator
Arrays and Objects	Creating arrays, array methods, loops, object properties, JSON basics	Product list
Functions	Function declarations, expressions, parameters, return values, arrow functions	Reusable calculator functions
Conditions and Loops	if/else, else-if, switch, ternary, for, while, do-while, break, continue	Grade checker and pattern tasks
DOM and Events	Selecting elements, changing text/styles, creating/removing elements, event listeners, mouse and keyboard events	Quiz app or memory game
Forms and Storage	Form validation, regex basics, date/time, localStorage, error handling with try/catch	To-do list with saved data
Async JavaScript	Callbacks, promises, async/await, API calls, loading states, error handling	Weather or movie search app
Final Project	Project planning, UI, logic, testing, debugging, presentation	Interactive web app or browser game

6.3 Python Programming

Module	Topics to Teach	Practice and Project
Setup and Basics	Python installation, PyCharm/VS Code setup, first project, print, variables, assignments	Hello Python and profile card
Data Types and Operators	Integers, floats, strings, booleans, built-in functions, math operations, comments	Grocery bill calculator
Input and Type Conversion	input(), int(), float(), str(), escape sequences, rounding	Celsius to Fahrenheit converter
Conditions	Comparison operators, if, else, elif, nested conditions, truthy/falsey values	Grade determiner
Loops	while loop, for loop, range, loop challenges, sum, factorial, Fizz Buzz	Number games
Strings and Collections	String methods, slicing, lists, tuples, word counter, character counter	Word counter app
Functions and OOP Basics	Functions, classes, objects, methods, __init__, special methods	Simple student management class
Advanced Basics	Decorators, generators, lambda, map, filter	Data processing mini exercises
File Handling	Read, write, append, CSV, JSON basics	Save and load a contact list
Final Project	Planning, functions, files, clean output, review	Console-based mini app or game

6.4 React Frontend Development

Module	Topics to Teach	Practice and Project
React Introduction	Why React, single page applications, environment setup, npm/npk/yarn, project structure	First React app
JSX and Components	JSX, functional components, folder structure, reusable UI blocks	Profile card components
Props and State	Passing data with props, useState, state updates, component re-rendering	Counter and product cards
Events and Rendering	Event handling, conditional rendering, lists, keys, destructuring	Dynamic product list
Forms	Controlled/uncontrolled components, validation, regex, React Hook Form basics	Registration form
Hooks	useState, useEffect, useRef, useCallback, useMemo, useContext, custom hooks	API data dashboard
Routing	React Router, navigation, nested routes, params, default route	Multi-page website
APIs and Error Handling	Axios/fetch, promises, async/await, loading states, try/catch, error boundaries	Search app with API
State and Optimization	Context API, Redux basics, thunk, memoization, lazy loading, Suspense	Dashboard or cart system
Testing and Deployment	Jest basics, component testing approach, Git/GitHub, Netlify/Vercel deployment	Live React capstone

6.5 Node.js, Express, and MongoDB Backend

Module	Topics to Teach	Practice and Project
Node.js Basics	Node overview, event-driven model, npm, package.json, modules, file/path/event modules	Basic Node app
Server Basics	Client-server architecture, HTTP server, Express introduction, routes, static files	Simple Express server
APIs	REST APIs, endpoints, GET/POST/PUT/DELETE, request/response objects	CRUD API
MongoDB and Mongoose	NoSQL, MongoDB setup, schemas, models, CRUD operations, data types, queries	Student or product database
Validation and Structure	Schema validation, custom validators, MVC pattern, routes, controllers, models	Organized backend project
Middleware and Security	Middleware types, custom middleware, error handling, API security basics	Protected routes
Authentication	Registration, login, bcrypt password hashing, JWT generation and verification	User authentication system
Authorization	Role-based access, authorization middleware, admin/user access	Admin panel backend
Advanced Database Work	Aggregation, projection, logical operators, SQL vs NoSQL basics	Reports API
Full-Stack Integration	Connect backend with React/React Native, Postman testing, deployment, interview prep	Task manager or e-commerce backend

6.6 Python Django Backend

Module	Topics to Teach	Practice and Project
Django Setup	Virtual environment, create project/app, project structure, first page	Hello Django website
Templates and Views	HTML integration, passing data from views to templates, static files, images	Dynamic pages
Models and ORM	Models, migrations, database tables, SQLite/PostgreSQL basics, ORM concepts	User information model
Forms and Views	Function-based views, class-based views, forms, validation, messages	Contact form
APIs and CRUD	GET, POST, update, delete APIs, filtering by name/email/date	User management API
Authentication	Signup, login, logout, dashboard, profile, login required decorators	User dashboard
Social Login	Google/GitHub/social authentication concept and setup approach	Social login demo
Email and OTP	Activation link, forgot password, email/OTP workflow	Secure account flow
AJAX and Interactions	AJAX requests, friend/follow request pattern, dynamic updates	Follow/request feature
Final Project	Planning, backend, templates/APIs, database, authentication, deployment basics	Blog, CRM, or student portal

6.7 Java Backend Development

Module	Topics to Teach	Practice and Project
Java Basics	JDK, JRE, JVM, setup, variables, data types, operators, comments, type casting	Basic Java programs
Control Flow	if/else, nested conditions, switch, loops, user input, functions/methods	Logic-building exercises
Core OOP	Classes, objects, constructors, this, static/non-static, inheritance, polymorphism, abstraction, encapsulation, interfaces	OOP mini project
Core Java Practice	Packages, access modifiers, arrays, strings, StringBuilder/StringBuffer, exceptions, date/time	Coding challenge set
Collections and Java 8	List, Set, Map, Queue, Stack, Iterator, Comparator, Comparable, streams, lambda, method references	Data handling programs
File I/O and JDBC	Byte/character streams, serialization, deserialization, JDBC connectivity, SQL basics	Database connected Java app
JSP and Servlet Basics	Servlet lifecycle, HTTP methods, request/response, sessions, cookies, filters, listeners, security basics	Login web app
Spring and Spring Boot	IoC, dependency injection, bean lifecycle, Spring MVC, Spring Boot app, properties, profiles, JPA, REST APIs	Spring Boot REST API
Hibernate and Microservices	ORM, mappings, sessions, transactions, HQL, caching, service boundaries, API gateway, cloud basics	Microservice-style backend demo
Interview and Capstone	Coding practice, project polishing, documentation, interview questions	Backend capstone with database

6.8 React Native Mobile App Development

Month	Topics to Teach	Project Output
1	HTML5 fundamentals, semantic structure, forms, validation, media, accessibility, responsive basics	Personal portfolio website
2	CSS3, box model, positioning, Flexbox, Grid, gradients, shadows, animations, Bootstrap/Tailwind overview	Company landing page
3	Modern JavaScript ES6, objects, arrays, promises, async/await, modules, DOM, local storage, API data	Interactive product catalog
4	Frontend integration, public APIs, debugging, browser tools, Git basics	E-commerce frontend prototype
5	React Native setup with Expo, project structure, JSX, components, state, props, hooks, navigation, mobile styling	Mobile news reader app
6	Context API, AsyncStorage, animations, camera/geolocation/notifications, forms with Formik/Yup, UX improvements	Fitness tracker mobile app
7	Node.js and Express backend, REST APIs, MongoDB/Mongoose, CRUD, JWT authentication, middleware, Postman	Task management backend
8	Full-stack integration, Context/Redux, environment variables, builds, backend deployment, app deployment planning	Full-stack mobile app capstone

7. Real-World Website, App, and Game Project Roadmap

Projects should be selected according to age, confidence level, and course track. Each project should include planning, coding, testing, improvements, and final explanation.

Website Projects

Level	Project Ideas	Skills Practiced
Beginner	My First Web Page, About Me Page, Hobby Page, Animal Page, Festival Page	HTML structure, text, images, links, basic styling
Intermediate	School Club Website, Restaurant/Cafe Website, Travel Landing Page, Event Website, Portfolio Website	CSS layouts, forms, responsive design, animations
Advanced	E-commerce Frontend, Blog Website, Course Landing Page, Admin Dashboard UI, Job Portal UI	JavaScript, React, routing, API data, deployment
Professional	Learning Management System, Booking Platform, Business Website with CMS-style sections, Full-Stack E-commerce	Frontend-backend integration, auth, database, hosting

App Projects

Level	Project Ideas	Skills Practiced
Beginner	Calculator, Grade Checker, Random Quote App, Simple Quiz App	Variables, conditions, functions, events
Intermediate	To-Do List, Expense Tracker, Weather App, Notes App, Movie Search App	DOM, local storage, API calls, async/await
Advanced	React Product Catalog, Task Manager, Dashboard, Chat UI, Blog App	React, hooks, routing, API integration
Professional	Full-Stack Task Manager, CRM, Fitness Tracker, News Reader, Food Order Mobile App	Backend APIs, authentication, mobile UI, deployment

Game Projects

Level	Project Ideas	Skills Practiced
Beginner	Animated Story, Maze Game, Catch Game, Simple Quiz Battle	Events, movement, loops, conditions, score
Intermediate	Tic-Tac-Toe, Memory Card Game, Typing Speed Game, Number Guessing Game, Hangman	Logic, arrays, functions, DOM, timers
Advanced	Snake Game, Platform Game, Space Shooter, Flappy-Style Game, Word Puzzle Game	Game loop, collision, keyboard events, state management
Challenge	Multilevel Quiz Game, Leaderboard Game, Educational Game with Local Storage	Data storage, UI design, scoring, replay logic

Project Rule for Every Batch

Every month should include at least one small project. Every major course should end with one polished capstone that has a title, problem statement, features list, screenshots, GitHub repository, and live demo where possible.

8. Teaching Methodology for Better Understanding

The teaching style should be simple, practical, and repeatable. Especially for school students, the teacher should not begin with definitions only. The class should begin with a familiar example, then move to code.

Recommended Class Flow

Step	What Teacher Does	Why It Helps Students
1. Warm-up	Ask one quick question or show the final output of the day.	Students know what they are going to build.
2. Explain Simply	Use real-life examples before technical words.	Concept becomes less scary and easier to remember.
3. Live Demo	Teacher codes slowly and explains every line.	Students see the thinking process, not just final code.
4. Guided Practice	Students code the same output with teacher support.	They gain confidence by completing a working task.
5. Mini Challenge	Give a small change: color, text, score, validation, layout, feature.	Students start thinking independently.
6. Debugging Time	Show common mistakes and how to read errors.	Students learn patience and problem solving.
7. Recap	Ask students to explain the concept in their own words.	Teachers confirm actual understanding.
8. Homework	Give one small task connected to the class project.	Practice continues without overwhelming the learner.

Age-Wise Teaching Notes

Age Group	Teaching Style	Avoid	Best Practice
Grades 3-5	Visual, story-based, playful, small steps, lots of encouragement	Long theory, heavy syntax, rushing to frameworks	Use characters, games, colors, and quick visible output
Grades 6-8	Practical examples, mini apps, challenges, pair discussion	Too much copy-paste or only lecture-based teaching	Let students customize projects with their own ideas
Grades 9-12	Concept clarity, structured coding, debugging, GitHub habits	Skipping fundamentals or ignoring code quality	Introduce planning, folder structure, reusable code, and deployment
College/Professionals	Outcome-focused, project-based, job-oriented, review-driven	Only syntax practice without real use cases	Use professional workflows, APIs, Git, deployment, and interview tasks

Teacher Checklist for Every Module

- Start with the final output and explain where the concept is used in real life.
- Give notes in simple English with one example and one practice question.
- Make students write code themselves instead of only watching the teacher.
- Keep a mistake board: common errors, reason, and solution.
- Review previous homework before starting a new topic.
- Maintain a project folder for each student and track progress weekly.
- End every module with a small presentation or code walkthrough.

9. Assessment, Certificates, and Portfolio System

Assessment should motivate students, not scare them. Younger students can be assessed through project demos, while older students and professionals should also complete quizzes, coding tasks, code reviews, and capstone presentations.

Assessment Type	Frequency	What to Check
Concept Recap	Every class	Can the student explain the concept in simple words?
Practice Task	Every class or alternate class	Can the student apply the concept without copying?
Mini Quiz	Every 2-3 weeks	Syntax, logic, definitions, debugging, and use cases
Mini Project	Every 4-6 weeks	Planning, code quality, completion, creativity, testing
Code Review	Monthly for older students	Naming, formatting, folder structure, repeated code, comments
Capstone Project	End of major course	Features, UI, functionality, deployment, presentation, documentation
Interview Practice	Career tracks	Technical questions, project explanation, problem solving, confidence

Certification Levels

Certificate	Suggested Requirement
Coding Explorer Certificate	Completion of beginner logic, block coding, and one creative game or story project.
Web Design Certificate	Completion of HTML/CSS course with one responsive website project.
JavaScript Developer Certificate	Completion of JavaScript basics, DOM, events, form validation, local storage, and one interactive web app.
Python Foundations Certificate	Completion of Python basics, loops, functions, collections, file handling, and one console project.
Frontend Developer Certificate	Completion of React modules with one deployed React project.
Backend Developer Certificate	Completion of Node/Django/Java backend with database and authentication project.
Full-Stack Developer Certificate	Completion of frontend, backend, database, deployment, GitHub, and capstone demo.
Mobile App Developer Certificate	Completion of React Native app with backend/API integration and final presentation.

10. Recommended Class Format and Delivery Plan

Because the academy is online, the structure should balance live explanation, coding practice, homework, and review. The same curriculum can run in regular, weekend, and fast-track formats.

Program Type	Class Frequency	Best For	Structure
School Regular Batch	2 classes per week	Grades 3-10	One concept class + one practice/project class each week

Program Type	Class Frequency	Best For	Structure
School Weekend Batch	1 longer class per week	Busy school students	Concept + guided practice + homework challenge
Senior Student Batch	2-3 classes per week	Grades 11-12	Coding + project + GitHub/deployment review
College Batch	3 classes per week	Internship/job preparation	Topic class + assignment + project review
Professional Fast-Track	3-5 classes per week	Working professionals	Focused modules, fewer basics, more project reviews
1:1 Mentoring	Flexible	Students needing personal attention	Customized roadmap, weekly milestones, direct feedback

Online Tools and Setup

Purpose	Recommended Tools
Live Classes	Zoom, Google Meet, Microsoft Teams
Code Editor	VS Code, Replit, CodePen, PyCharm for Python, IntelliJ/Eclipse for Java
Design and Planning	Figma basics, Canva for simple assets, whiteboard tools, flowcharts
Version Control	Git and GitHub for Grades 9+, college, and professional tracks
API Testing	Postman for backend and full-stack batches
Databases	MongoDB, SQLite, PostgreSQL/MySQL based on track
Deployment	Netlify, Vercel, Render, GitHub Pages, and mobile build tools where appropriate
Communication	WhatsApp/Email/LMS for homework, feedback, recordings, and announcements

Parent and Student Communication Plan

- Share monthly progress updates for school students with completed topics, project screenshots, and next goals.
- Keep a simple student progress sheet: attendance, homework, project completion, quiz score, and teacher remarks.
- Run a demo day every 2-3 months where students present what they built.
- For career tracks, share GitHub progress, project links, interview practice status, and improvement notes.

Suggested Course Bundles for Kaur's Coding Academy

Bundle Name	Audience	Duration	Includes	Final Outcome
Junior Coding Explorer	Grades 3-5	4-6 months	Computer basics, logic, block coding, beginner web, Python fun, games	Creative game/story + first web page
Creative Web and Games Builder	Grades 6-8	6-9 months	HTML, CSS, JavaScript, DOM, Python basics, browser games	Responsive website + interactive game/app
Teen Developer Program	Grades 9-10	8-10 months	Web design, JavaScript, APIs, GitHub, React basics, project sprint	Portfolio + API-based web app
Career Starter Program	Grades 11-12	10-12 months	React, backend basics, database, authentication, deployment, capstone	Full-stack project with GitHub and live demo

Bundle Name	Audience	Duration	Includes	Final Outcome
Full-Stack Web Developer	College/Professionals	7-9 months	HTML/CSS/JS, React, Node.js, Express, MongoDB/SQL, Git, deployment, interviews	Job-ready full-stack portfolio
Mobile App Developer	College/Professionals	8 months	Frontend foundation, React Native, Node.js, MongoDB, APIs, app deployment planning	Full-stack mobile app
Backend Developer	College/Professionals	5-10 months	Python/Django or Java/Spring Boot, databases, APIs, authentication, deployment	Backend API capstone
Professional Upskilling Fast-Track	Professionals	12-20 weeks	Customized React, Node, Django, Java, or React Native path with project reviews	Production-style project and interview confidence

Final Positioning Statement

Kaur's Coding Academy can be positioned as an online project-based coding academy for school students, college students, and professionals, with clear age-wise learning paths and career-focused advanced tracks. The strongest selling point is not only teaching programming, but helping learners build real-world websites, apps, and games step by step.

Final Notes for Implementation

The roadmap should be used as a flexible framework. The teacher can slow down for younger students, add extra practice for weak batches, or compress the syllabus for professionals. The most important rule is consistency: every topic should connect to a practical output, and every learner should finish with a project portfolio.

Quick Start Plan for Launch

1. Start with three flagship programs: Junior Coding Explorer, Creative Web and Games Builder, and Full-Stack Web Developer.
2. Create a simple entrance form to identify the student grade, previous coding experience, and goal.
3. Run a free demo class based on age group: game demo for kids, website demo for teens, full-stack roadmap demo for college/professionals.
4. Maintain a project gallery for student work and update it every month.
5. Use monthly demo days as the academy's strongest proof of learning.