---

1. What the app *is* (high-level)

From the screens, your app is:

A training platform for industrial automation where users can register, browse courses (PLC, SCADA, HMI, Robotics, etc.), consume lessons & live classes, track progress, receive certificates, and admins can manage users & courses.

Key roles

- **Learner**
- **Admin** (course & user management)

---

2. Screens & flows (based directly on the HTML)

I'll list each screen (using folder name + page title) and describe what it should do in the real app.

2.1 Onboarding / Marketing

1) welcome_screen_1 – *Automation Training App*

- Headlines like: "Master Automation & Control Systems", "Expert-Led Courses", "Interactive Simulations", "Learn Anywhere".
- **Buttons**: Create Account, Sign In.
- **Flow**:
    - Create Account → registration_start_screen_* or create_account_screen.
    - Sign In → login_screen.

2) welcome_screen_2 – *Automation Training App*

- Variant hero screen: "Master Industrial Automation", "Hands-On Sims", "Career Growth".
- **Button**: Get Started → same as above (registration entry point).

3) about/contact_us_screen – *About Our Center*

- Headings: "About Our Center", "Pioneering Automation Education", "Get in Touch".
- Static info about the training center + contact details (email/phone/address/form).
- **Flow**: reachable from nav/footer, no heavy logic.

2.2 Registration & Authentication

5) registration_start_screen_2 – *Get Started - Automation Training*

- Headline: "Get Started".
- **Buttons**:
    - Register with Email

  - "Continue with Google"
  - "Continue with Apple` (based on the SVGs/buttons).
- **Flow**:
  - Email option → create_account_screen.
  - Social options → OAuth flow (back-end endpoints for social login).

6) create_account_screen – *Create Account*

- Heading: "Create Your Account".
- Inputs:
  - Email (you@email.com)
  - Password (Enter a secure password)
  - Confirm password (Re-enter your password)
- Button: Create Account.
- **Flow**:
  - POST registration → if success → registration_success_screen or profile_setup_screen_3.

7) registration_success_screen – *Registration Successful*

- Heading: "Registration Successful!".
- Button: Explore Courses.
- **Flow**: go to progress_dashboard or course_catalog_1.

8) login_screen – *Login - Automation Training*

- Heading: "Welcome Back".
- Inputs:
  - Username/email (Enter your username or email)
  - Password (Enter your password) + eye icon (visibility) to toggle.
- Button: Login.
- May contain links like Forgot password? → forgot_password_request.
- **Flow**:
  - POST login → if success → progress_dashboard_1.

9) forgot_password_request – *Reset Password*

- Heading: "Reset Password".
- Back icon.
- Input: probably email ("Enter your email address", visible in HTML).
- Button: Send Reset Link.

- **Flow**:
    - POST request → generates token, sends email → show password_reset_confirmation.

10) password_reset_confirmation – *Check Your Email*

- Heading: "Check Your Email".
- Explanation: reset link sent.
- Maybe Back to Login button.
- **Flow**: from here user uses emailed link → new_password_entry_screen.

11) new_password_entry_screen – *Set New Password*

- Heading: "Create New Password".
- Inputs:
    - New password.
    - Confirm password.
- There's a checklist under the password field:
    - "At least 8 characters"
    - "1 uppercase letter"
    - "1 number"
    - "1 special character (!@#$...)"
- Button: Save Password / "Continue".
- **Flow**:
    - POST new password with token → success → either auto-login or go back to login_screen.

2.3 Profile & Settings

12) profile_setup_screen_3 – *Create Your Profile*

- Heading: "Create Your Profile".
- Inputs:
    - Full Name (John Doe)
    - Email (you@example.com)
    - Likely dropdowns/checkboxes for role/experience.
- Buttons/Chips for areas of interest:
    - PLC, SCADA, Robotics, etc. (visible as interest chips in HTML).
- **Flow**:
    - Save profile → server updates UserProfile & LearningPreferences.

    - Next → recommended_courses or progress_dashboard.

13) edit_user_profile_screen – ***Edit Profile - Automation Training***

- Heading: “Edit Profile”.
- Shows:
    - Name (e.g. “Amelia Clarke”).
    - Email (read-only, amelia.c@example.com).
- Section: “Learning Interests” with chips:
    - PLC Programming, SCADA Systems, HMI Design, Automation, Robotics, etc.
- **Flow**:
    - Toggle chips → PATCH to API.
    - Save button → update UserProfile.

14) notification_settings_screen – ***Notification Preferences***

- Heading: “Notification Preferences”.
- Sections:
    - “Course Alerts”
    - “Content & Community”
    - “General”
- Each section likely contains toggle switches (email / push / SMS).
- **Flow**:
    - Update toggles → API to update NotificationSettings.

2.4 Course Discovery & Recommendations

15) course_catalog_1 – ***Course Catalog - Automation Training***

- Heading: “Courses”.
- Shows list of course cards like:
    - PLC Programming
    - “Introduction to Siemens S7-1200”
    - “Allen-Bradley CompactLogix Basics”
    - SCADA Systems, etc.
- Top icons:
    - Notification bell (notifications)
    - Filter / Tune icon (tune) for advanced filters.

- **Flow**:
  - Tap course card → course_details_screen_*.
  - Filter → open filters modal.
  - Notification icon → maybe notifications page (not in this pack).

16) course_catalog_2 – *Course Catalog - Automation Training*

- Alternate catalog layout:
  - Has search bar ("Search for courses…") & maybe categories.
- **Flow** similar to above.

17) recommended_courses – *Course Recommendations*

- Heading: "Recommended For You".
- Sub-sections:
  - "Because You Completed Intro to PLC"
  - "New in SCADA"
  - "Top Picks For You"
- Actions:
  - View Details for each recommended course.
  - Start Learning on one of them.
- **Flow**: uses user's history & interests → backend recommended courses.

2.5 Course Details, Learning, Certificates

18) course_details_screen_1 – *Course Details*

- Headings:
  - "Course Details"
  - "Advanced PLC Programming"
  - "Instructor"
  - "Learning Objectives"
  - "Syllabus"
- Buttons:
  - Back arrow
  - Bookmark (bookmark_border)
  - Enroll Now
- Cards/sections:

    - Duration, Level, Rating, etc. (from HTML badges).
    - "Learning Objectives" bullet list.
    - "Syllabus" – list of modules/lessons.
    - "Instructor" – name, photo, bio.
- **Flow**:
    - Enroll → create Enrollment → redirect to learning_module_screen or progress_dashboard.

19) course_details_screen_2 – ***Course Details***

- Same course (Advanced PLC Programming) but with tabbed layout:
    - Tabs: About, Syllabus, Instructor.
    - Share icon.
    - Enroll Now button.
- **Flow**:
    - Tab switch just changes section.
    - Enroll + share course.

20) learning_module_screen_1 – ***Course Content Delivery***

- For a specific lesson:
    - Lesson title ("Intro to PLC", "Lesson 5: Understanding Ladder Logic").
    - "Key Concepts" (bullet points).
    - "Knowledge Check" – quiz area.
- Buttons:
    - Navigation arrows (arrow_back_ios, arrow_forward_ios).
    - Quiz options are buttons: e.g. "To connect to an external display".
- **Flow**:
    - Render video/text content.
    - User answers quiz → POST responses → mark lesson complete & update progress.
    - Next/Previous lesson navigation.

21) learning_module_screen_2 – ***Course Content Delivery***

- Similar layout, another lesson (Module 1: Intro to PLC).
- Contains multiple-choice questions under "Knowledge Check".
- Buttons/choices like:
    - "To provide physical power"
    - "To represent a single line of logic"

- **Flow** same as above.

22) certificate_of_completion – *Certificate of Completion*

- Heading: “Certificate of Completion”.
- Shows user name, course name, completion date.
- Buttons:
    - Download (download icon)
    - Share
- **Flow**:
    - PDF generation on backend.
    - Download (PDF file).
    - Share (email/link/social).

2.6 Dashboards & Live Content

23) progress_dashboard_1 – *Dashboard - Automation Training*

- Headings:
    - “Overall Progress”
    - “My Courses”
- Elements:
    - User avatar.
    - Progress bar/percentage across courses.
    - List of ongoing courses with progress (e.g. 40% complete) and actions:
        - Resume Learning
        - Review Course
- **Flow**:
    - API returns aggregated learner stats.
    - Links to learning_module_screen or course_details_screen.

24) progress_dashboard_2 – *Dashboard - Automation Training*

- Variant dashboard layout:
    - Possibly adds “Upcoming Sessions”, “Recommended Next Course”.
- Buttons:
    - Join Live, Watch Now, etc.
- **Flow**:

    - Mix of ongoing courses and live events.

25) profile_setup_screen_1 – *Live Classes & Webinars*

- Title: “Live Classes & Webinars”.
- Sections:
    - “Upcoming Live Sessions” – list of sessions (e.g. “Introduction to SCADA Security”, “HMI Design Best Practices”).
- Buttons: Join Session, Set Reminder, Watch Now etc.
- **Flow**:
    - Session objects with date/time, link/meeting_info.
    - Setting reminder → persists in UserLiveSessionPreference.

2.7 Admin Area

26) profile_setup_screen_2 – *Admin Course Management*

- Title: “Admin Course Management”.
- Heading: “Course Management”.
- Cards:
    - “Introduction to PLC Systems”
    - “Automation with Python”
    - “Industrial Robotics Fundamentals”
- Buttons:
    - Add (add_circle)
    - Edit
- **Flow**:
    - Admin creates/edits courses, sets metadata, toggles publish status.

27) admin_user_management – *User Management*

- Title & heading: “User Management”.
- Search bar: “Search by name or email…”.
- User list with 3-dots menu (more_vert) for actions.
- **Flow**:
    - Admin filters/searches for users.
    - Actions likely: view profile, reset password, deactivate/activate.

---

3. Backend Design (Spring Boot)

Here’s a backend blueprint matching those screens.

3.1 Core Entities (JPA)

User

- id
- email (unique)
- passwordHash
- fullName
- role (ENUM: LEARNER, ADMIN)
- createdAt, updatedAt
- status (ACTIVE, SUSPENDED)

UserProfile

- id
- user (OneToOne)
- headline / jobTitle
- experienceLevel (BEGINNER, INTERMEDIATE, ADVANCED)
- company (optional)
- bio
- interests (many-to-many with Tag or string list)

NotificationSettings

- id
- user (OneToOne)
- courseAlertsEmail (boolean)
- courseAlertsPush
- contentCommunityEmail
- contentCommunityPush
- generalEmail
- generalPush

Course

- id
- title (e.g. “Advanced PLC Programming”)
- shortDescription

- fullDescription / about
- level (BEGINNER, INTERMEDIATE, ADVANCED)
- durationMinutes
- category (PLC, SCADA, HMI, Robotics, etc.)
- thumbnailUrl
- isPublished
- createdBy (Admin user)

Module

- id
- course (ManyToOne)
- title
- orderIndex

Lesson

- id
- module (ManyToOne)
- title
- contentType (VIDEO, ARTICLE, SCORM, etc.)
- contentUrl or contentHtml
- durationMinutes
- orderIndex

Quiz

- id
- lesson (OneToOne or ManyToOne)
- title

Question

- id
- quiz
- text
- type (SINGLE_CHOICE, MULTI_CHOICE)
- orderIndex

AnswerOption

- id

- question
- text
- isCorrect

Enrollment

- id
- user
- course
- status (ENROLLED, COMPLETED, DROPPED)
- enrolledAt
- completedAt

LessonProgress

- id
- enrollment
- lesson
- status (NOT_STARTED, IN_PROGRESS, COMPLETED)
- lastViewedAt
- score (quiz score if any)

LiveSession

- id
- title
- description
- course (optional)
- startTime
- endTime
- joinUrl
- recordingUrl
- isUpcoming

LiveSessionReminder

- id
- user
- liveSession
- remindAt

Certificate

- id
- user
- course
- issuedAt
- certificateNumber
- pdfUrl

Recommendation

- id
- user
- course
- reason (string, e.g. “Because you completed Intro to PLC”)
- source (ALGO, MANUAL)

3.2 REST API Outline

AuthController

- POST /api/auth/register
- POST /api/auth/login
- POST /api/auth/logout
- POST /api/auth/refresh-token
- POST /api/auth/forgot-password
- POST /api/auth/reset-password

UserController

- GET /api/users/me
- PATCH /api/users/me
- GET /api/users/me/profile
- PATCH /api/users/me/profile
- GET /api/users/me/notification-settings
- PATCH /api/users/me/notification-settings

CourseController

- GET /api/courses
    - Query params: search, category, level, sort.

- GET /api/courses/{courseId}
- GET /api/courses/{courseId}/modules
- GET /api/courses/{courseId}/modules/{moduleId}/lessons
- GET /api/lessons/{lessonId}

EnrollmentController

- POST /api/courses/{courseId}/enroll → used by Enroll Now.
- GET /api/enrollments/my → “My Courses” section (dashboard).
- GET /api/enrollments/{enrollmentId}/progress → detailed progress.

ProgressController

- POST /api/lessons/{lessonId}/start
- POST /api/lessons/{lessonId}/complete
- POST /api/quizzes/{quizId}/submit → returns score, correctness.
- GET /api/dashboard/overview → “Overall Progress” for learner.

RecommendationController

- GET /api/recommendations → for recommended_courses screen.

CertificateController

- GET /api/certificates/my → list.
- GET /api/certificates/{id} → details.
- GET /api/certificates/{id}/download → returns PDF file.

LiveSessionController

- GET /api/live-sessions/upcoming
- POST /api/live-sessions/{id}/reminders (user sets reminder)
- POST /api/live-sessions/{id}/join (returns join URL / token)

AdminCourseController` (ROLE_ADMIN)

- GET /api/admin/courses
- POST /api/admin/courses
- PUT /api/admin/courses/{id}
- DELETE /api/admin/courses/{id}
- POST /api/admin/courses/{id}/publish / unpublish

AdminUserController` (ROLE_ADMIN)

- GET /api/admin/users?search=...
- GET /api/admin/users/{id}

- PATCH /api/admin/users/{id} (role, status)
- POST /api/admin/users/{id}/reset-password (admin-triggered reset)

This API set is enough to support every visible button and screen in your UI.

---

4. Flutter App Structure

4.1 Suggested folder structure

```
lib/
  main.dart
  core/
    config/
    theme/
    routing/
    widgets/
    network/
  features/
    auth/
      data/
      domain/
      presentation/
    profile/
    courses/
    learning/
    dashboard/
    admin/
    live_sessions/
```

Use something like **Riverpod** or **Bloc** for state management.

4.2 Routes → Screens mapping

Roughly:

- / → WelcomeScreen (welcome_screen_1 / 2)
- /auth/login → LoginScreen
- /auth/register-start → RegistrationStartScreen2
- /auth/create-account → CreateAccountScreen

- /auth/forgot-password → ForgotPasswordScreen
- /auth/reset-password → NewPasswordScreen
- /onboarding/profile → CreateProfileScreen (profile_setup_screen_3)
- /dashboard → DashboardScreen (progress_dashboard_1/2)
- /courses → CourseCatalogScreen
- /courses/:id → CourseDetailsScreen
- /courses/:courseId/module/:moduleId/lesson/:lessonId → LessonScreen (learning_module_screen_1/2)
- /recommended → RecommendedCoursesScreen
- /notifications → NotificationSettingsScreen
- /profile/edit → EditProfileScreen
- /live-sessions → LiveSessionsScreen (profile_setup_screen_1)
- /certificates/:id → CertificateScreen
- /admin/courses → AdminCourseManagementScreen
- /admin/users → AdminUserManagementScreen
- /about → AboutCenterScreen

Each Flutter screen mirrors the layout in the HTML: cards, chips, tabs, lists, etc.

4.3 Example: Flutter – model for Course & simple service

```
class Course {
  final String id;
  final String title;
  final String shortDescription;
  final String level;
  final int durationMinutes;
  final String thumbnailUrl;
  final String category;
  final bool isEnrolled;

  Course({
    required this.id,
    required this.title,
    required this.shortDescription,
    required this.level,
    required this.durationMinutes,
    required this.thumbnailUrl,
    required this.category,
    required this.isEnrolled,
  });

  factory Course.fromJson(Map<String, dynamic> json) {
    return Course(
      id: json['id'],
      title: json['title'],
      shortDescription: json['shortDescription'],
      level: json['level'],
      durationMinutes: json['durationMinutes'],
      thumbnailUrl: json['thumbnailUrl'],
      category: json['category'],
      isEnrolled: json['isEnrolled'] ?? false,
    );
  }
}
class CourseApi {
  final Dio _dio;

  CourseApi(this._dio);

  Future<List<Course>> getCourses({String? search}) async {
    final response = await _dio.get(
      '/api/courses',
      queryParameters: {'search': search},
    );
    final list = (response.data as List)
        .map((e) => Course.fromJson(e as Map<String, dynamic>))
      .toList();
    return list;
  }

  Future<Course> getCourseDetails(String id) async {
    final response = await _dio.get('/api/courses/$id');
    return Course.fromJson(response.data);
  }

  Future<void> enrollInCourse(String id) async {
    await _dio.post('/api/courses/$id/enroll');
  }
}
```
