Modules :

- **Authentication & Authorization (`Auth` Module)**: Manages user identity, registration, login, roles, and permissions.
- **Users (`Users` Module)**: Handles user profiles, interests, relationships (parent-child), and settings.
- **Content (`Content` Module)**: The core module for managing all educational content: Categories, Learning Paths, Courses, and Grades.
- **Commerce (`Commerce` Module)**: Manages financial transactions, including Carts, Packages, Coupons, and Payments.
- **Engagement (`Engagement` Module)**: Fosters community interaction through Feeds, Comments, and Ratings.
- **Productivity (`Productivity` Module)**: Includes personal management tools like the Calendar and daily tracking.
- **Notifications (`Notifications` Module)**: A centralized service for handling all email and in-app notifications.
- **AI Services (`AI` Module)**: A dedicated module for all artificial intelligence features.

1. **Authentication & Authorization (`Auth`) Module**
    1. **Entities**
        - `User` **Schema:**
            - `_id` : `ObjectId`
            - `name` : `String` , required
            - `email` : `String` , required, unique, indexed
            - `password` : `String` , required (hashed)
            - `country` : `String` , required
            - `phoneNumber` : `String` , required, unique
            - `role` : `Enum ['student', 'parent', 'admin' , 'content_creator']` , required, indexed
            - `isEmailVerified` : `Boolean` , default: `false`
            - `googleId` : `String` , optional, unique, sparse index
            - `firstLogin` : `Boolean` , default: `true`
            - `studentProfile` : `Object` , (conditional, if role is 'student')
                - `dob` : `Date` , required
                - `religion` : `String` , required
                - `parent` : `ObjectId` , ref: 'User', optional
                - `daily usage limit`

        - `elasped time of the day`
    - `parentProfile`: `Object`, (conditional, if role is 'parent')
        - `religion`: `String`, required
        - `children`: `[ObjectId]`, ref: 'User'
    - `timestamps`: `createdAt`, `updatedAt`
    - `Interests` (tags/keywords)
    - `Points` (for rewards system)
2. **Services**
    - **Registration Service**:
        - Validates user input.
        - Hashes the password.
        - Creates a new `User` document.
        - Generates a unique email verification token.
        - Dispatches a "Confirm Email" event to the **Notifications Module**.
    - **Login Service**:
        - Validates credentials (email/password).
        - Generates JWT `accessToken` and `refreshToken`.
        - Checks the `firstLogin` flag. If true, dispatches a "Welcome Email" event and sets the flag to `false
    - **OAuth Service (Google)**:
        - Handles Google Sign-In/Sign-Up callback.
        - Finds an existing user by `googleId` or `email`, or creates a new one.
        - Generates JWTs.
    - **Password Management Service**:
        - Handles "forgot password" requests by sending a reset link.
        - Verifies reset tokens and updates the user's password.`
3. **API Endpoints**
    - `POST /auth/register`: Register a new user.
    - `POST /auth/login`: Log in and receive JWTs.
    - `POST /auth/google`: Authenticate with Google.
    - `GET /auth/verify-email?token=<token>`: Verify user's email via link.
    - `POST /auth/resend-verification`: Resend the verification email.
    - `POST /auth/forgot-password`: Request a password reset link.
    - `POST /auth/reset-password`: Set a new password using a reset token.

---

2. **Users (`Users`) Module**
    1. **Entities**
        - `User` **Schema:** (As defined in Auth Module)
        - `Interest` **Schema:**
            - `_id`: `ObjectId`
            - `name`: `String`, unique
            - `description`: `String`
            - `targetAudience`: `Enum ['student', 'parent', 'all']`, indexed
        - **User-Interest Relationship:**
            - A `interests` field in the `User` schema: `interests: [ObjectId]`, ref: 'Interest'.
    2. **Services**
        - **Profile Management Service**:
            - Allows users to fetch and update their profile information.
            - Handles logic for sending a new verification if the phone number is changed.
            - Manages linking/unlinking parent accounts (updates `parent` field in student and `children` field in parent).
        - **Interest Service**:
            - Fetches interests based on user role (`student` vs. `parent`).
            - Assigns selected interests to the user profile.
    3. **API Endpoints**
        - `GET /users/me`: Get the profile of the currently logged-in user.
        - `PUT /users/me`: Update the user's profile (name, country, etc.).
        - `PUT /users/me/password`: Change the user's password (requires current password).
        - `GET /users/interests`: Get a list of available interests based on user role.
        - `PUT /users/me/interests`: Update the user's selected interests.
        - `POST /users/me/link-parent`: (For students) Send a request to a parent's email to link accounts.
        - `GET /users/kids-dashboard`: (For parents) Get an aggregated dashboard of their children's progress.

---

3. **Content (`Content`) Module**
    1. **Entities**
        - `Category` **Schema:**
            - `name`: `String`, required, unique

    - `description`: `String`
    - (Indexes on `name` and `description` for text search)
- `LearningPath` **Schema:**
    - `title`: `String`, required
    - `description`: `String`
    - `category`: `ObjectId`, ref: 'Category', indexed
    - `price`: `Number`
    - `totalHours`: `Number`
    - `language`: `String`
    - (Indexes on `title`, `description`, `category`, `price`, `totalHours`, `language`)
- `Course` **Schema:**
    - `title`: `String`, required
    - `description`: `String`
    - `learningPath`: `ObjectId`, ref: 'LearningPath', indexed
    - `presenter`: `ObjectId`, ref: 'User' (Content Creator role)
    - `totalHours` : Float
    - `prerequisites: [ObjectId]`,
    - `outcome: [String]`,
    - `language` : String
    - `targetAudience` : String
    - `ageRange` : `[number]`
        - `coverImage`: `String` (URL to S3/Cloud Storage)
        - `price`: `Number`
        - `chunks`: `[ChunkSchema]`
        - `attachments`: `[FileSchema]`
        - `averageRating`: `Number`, default: 0
    - QualificationExam : 0bjectId
    - FinalExam : 0bjectId
    - Certificate : 0bjectId
    - NumberOfLessons : int
    - `Type` (Standalone, LearningPathCourse)
    - `DifficultyLevel` (Beginner, Intermediate, Advanced)
    - `IsLocked` (Boolean)
    - Organization : ObjectId
    - `Status` (Draft, Published, Archived)
    - `DisplayMode` (LearningPath, Grade)

    - `CreatedDate`
    - `UpdatedDate`
- `Chunk` **(Sub-document) Schema:**
    - `title`: `String`
    - `type`: `Enum ['video', 'audio', 'text']`
    - `contentUrl`: `String`
    - `transcript_en`: `String`
    - `transcript_ar`: `String`
    - `quiz`: `ObjectId`, ref: 'Quiz'
    - `lab`: `Object` (Lab details)
    - `OrderIndex` (sequence inside the course)
    - `EstimatedDuration` (minutes)
    - `RequiresLab` (Boolean)
- `Quiz`, `Exam`, `Assignment`, `Certificate`, `Note` **Schemas:** Separate collections linked to `User` and `Course`.
- `UserCourseProgress` **Schema:** (Tracks user progress)
    - `user`: `ObjectId`, ref: 'User'
    - `course`: `ObjectId`, ref: 'Course'
    - `completedChunks`: `[ObjectId]`
    - `progressPercentage`: `Number`
    - `lastVisited`: `Date`
    - `points`: `Number`
    - `notes`: `[NoteSchema]` (Notes taken by the user for this course)
    - (Compound index on `user` and `course`)

2. **Services**
    - **Course Discovery Service**:
        - Handles fetching, searching, filtering (by category, price, etc.), and sorting of categories, paths, and courses.
        - Implements pagination for all list-based endpoints.
        - Integrates with the **AI Module** to show personalized suggestions first.
    - **Course Enrollment & Access Service**:
        - Checks for prerequisites or qualification exam completion before granting access.
        - Creates/updates the `UserCourseProgress` document upon enrollment.
    - **Progress Tracking Service**:
        - Updates user progress as they complete chunks and quizzes.
        - Awards points.

- Triggers certificate generation upon course completion and exam pass.

3. **API Endpoints**
   - `GET /categories`: List all categories (paginated).
   - `GET /learning-paths`: List all learning paths (paginated, with filters for category, price, etc.).
   - `GET /courses`: List all courses (paginated, with filters).
   - `GET /courses/:id`: Get detailed information for a single course.
   - `POST /courses/:id/enroll`: Enroll in a course (payment is handled by Commerce module).
   - `GET /courses/:id/progress`: Get current user's progress for a course.
   - `POST /courses/:courseId/chunks/:chunkId/complete`: Mark a chunk as complete.
   - `POST /courses/:id/notes`: Add a note to a course.
   - `GET /my-learning`: Get a list of all courses the user is enrolled in.

---

4. **Commerce (`Commerce`) Module**
   1. **Entities**
      - `Cart` **Schema:**
        - `user`: `ObjectId`, ref: 'User', unique
        - `items`: `[{ course: ObjectId, ref: 'Course' }]`
        - `totalPrice`: `Number`
      - `Coupon` **Schema:**
        - `code`: `String`, unique, indexed
        - `discountPercentage`: `Number`
        - `expiryDate`: `Date`
        - `isActive`: `Boolean`
      - `Order` **Schema:**
        - `user`: `ObjectId`, ref: 'User'
        - `courses`: `[ObjectId]`, ref: 'Course'
        - `amount`: `Number`
        - `paymentStatus`: `Enum ['pending', 'completed', 'failed']`
        - `transactionId`: `String`
      - `Package` **Schema:**
        - `name`: `String`
        - `courses`: `[ObjectId]`, ref: 'Course'
        - `price`: `Number`

2. **Services**
    - **Cart Service**: Adds, removes, and lists items in a user's cart. Recalculates total price on modification.
    - **Coupon Service**: Validates coupon codes and applies discounts to the cart total.
    - **Payment Service**:
        - Integrates with a third-party payment gateway (e.g., Stripe, PayPal).
        - Creates a payment intent/session.
        - Handles payment confirmation webhooks to create an `Order` and enroll the user in the purchased courses.
    - **Stared/Favorites Service**: Manages a user's list of favorite courses (`staredCourses: [ObjectId]` field in `User` schema).
3. **API Endpoints**
    - `GET /cart`: Get the user's current cart.
    - `POST /cart`: Add a course to the cart.
    - `DELETE /cart/items/:courseId`: Remove a course from the cart.
    - `POST /cart/apply-coupon`: Apply a coupon to the cart.
    - `POST /checkout`: Initiate the payment process.
    - `GET /stared`: Get the user's list of stared courses.
    - `POST /stared`: Add a course to the stared list.
    - `DELETE /stared/:courseId`: Remove a course from the stared list.

---

5. Engagement (`Engagement`) Module

1. **Entities**
    - `Feed` **Schema:**
        - `author`: `ObjectId`, ref: 'User'
        - `content`: `String`
        - `media`: `[{ type: Enum['image', 'video'], url: String }]`
        - `visibility`: `Enum ['public', 'private', 'specific_users']`
        - `allowedUsers`: `[ObjectId]`, ref: 'User'
    - `Comment` **Schema:**
        - `author`: `ObjectId`, ref: 'User'
        - `content`: `String`
        - `parent`: `ObjectId` (can refer to a `Course`, `Feed`, or another `Comment` for replies)
        - `isHighlighted`: `Boolean`

- `Like` **Schema:**
  - `user` : `ObjectId` , ref: 'User'
  - `target` : `ObjectId` (can refer to a `Feed` or `Comment`)
  - (Compound unique index on `user` and `target`)

2. **API Endpoints**
   - `GET /feeds` : Get a public feed timeline (paginated).
   - `POST /feeds` : Create a new feed post.
   - `DELETE /feeds/:id` : Delete a user's own feed post.
   - `GET /courses/:id/comments` : Get comments for a course.
   - `POST /courses/:id/comments` : Add a comment to a course.
   - `POST /feeds/:id/like` : Like/unlike a feed post.

---

6. AI Services (`AI`) Module

1. **Services (No entities, acts as a proxy)**
   - **Recommendation Service**:
     - Input: `userId` , `interests` , `age` , `viewedCourses` .
     - Logic: Uses collaborative filtering or content-based models to generate a ranked list of `categoryId` , `learningPathId` , and `courseId` .
     - Output: `[ { recommendationId, score } ]` .
   - **Content Generation Service**:
     - `generateQuiz(text, difficulty)` : Creates a quiz from course content.
     - `generateTranscript(mediaUrl)` : Generates AR/EN transcripts.
     - `generateDescription(content)` : Generates a summary/description.
   - **Search Service**:
     - `audioSearch(audioClip)` : Converts audio to text and performs a search.
   - **Text-to-Speech Service**:
     - `synthesizeSpeech(text, voice)` : Converts text to audio.
2. **API Endpoints (Internal or External)**
   - `GET /ai/recommendations` : Get personalized content recommendations for the current user.
   - `POST /ai/generate-quiz` : (Used internally by course creators) Generate a quiz.
   - `POST /ai/generate-transcript` : (Used internally) Generate a transcript for a video/audio chunk.
   - `POST /ai/search` : Perform a search (can handle text or audio multipart upload).