**Comprehensive Implementation Plan for TalentHíd – Tanulás élethosszig 2.0**

**1. Overview**

TalentHíd is a hybrid learning platform designed to offer synchronous modular courses on a fixed day (e.g., Wednesdays). The system supports:

* **Individual registration:** Each user registers with an email and password, confirms via a confirmation link, and then logs in to access courses.
* **Training Room Mode:** Institutions (e.g., local learning centers) register as “training rooms.” In these scenarios, a training room account is created and later, when students use the shared computer, the system prompts for an additional login so that each student’s personal learning materials are correctly associated.
* **Course Booking & Scheduling:** Users book courses in advance. The system implements an intelligent scheduling logic to prevent overlapping live sessions while also allowing bookings of recorded content to fill in personal training schedules.
* **Live Streaming & Recording:** A self-hosted streaming solution (using WebRTC/RTMP via NGINX-RTMP) streams live sessions to up to 800 concurrent users. Sessions are automatically recorded, and only registered participants can access the recordings.
* **Personal Learning Folder:** Every registered user has a personal “learning folder” where they can access course materials (PDFs, audio files, etc.) that are versioned and only available for the courses they booked.
* **Administrative Panel:** An integrated administration and reporting module enables course, user, and content management along with role-based access and detailed reporting.

The entire system is built as a responsive, mobile-optimized Progressive Web App (PWA) designed for a maximum resolution of 1024×768 pixels. The backend is implemented in Node.js using TypeScript with frameworks like Express.js or NestJS, and the frontend is developed using React.js (or Next.js if server-side rendering is needed). PostgreSQL is used for relational data storage, Redis for caching, and Socket.io for real-time communication.

**2. Detailed Workflows and Scenarios**

**2.1. User Registration and Authentication Workflows**

**2.1.1. Standard User Registration**

1. **User Initiation:**
   * The user navigates to the registration page.
   * A form collects the following information:
     + Email address
     + Full name
     + Password (with strength validation)
2. **Backend Processing:**
   * **Endpoint:** POST /api/register
   * The backend validates the form data.
   * The password is hashed using bcrypt.
   * A new user record is inserted into the PostgreSQL database with fields: id, email, name, hashed\_password, and an isActive flag set to false.
   * A unique token is generated (e.g., a JWT or UUID) and stored in a temporary confirmation table (or as a field in the user record).
   * A confirmation email is sent to the user with a link containing the token.
3. **Email Confirmation:**
   * **Endpoint:** GET /api/confirm?token=<token>
   * When the user clicks the link, the backend verifies the token.
   * If valid, the user’s isActive flag is set to true.
   * The user is redirected to the login page with a success message.
4. **User Login:**
   * **Endpoint:** POST /api/login
   * User submits email and password.
   * Backend verifies credentials and returns a JWT token.
   * The token is stored in the client’s session (or local storage) for authentication of subsequent requests.

**2.1.2. Training Room Registration and Student Additional Login**

1. **Training Room Registration (by an Administrator):**
   * A special “Training Room” registration page is provided.
   * The administrator enters institution details (e.g., name, location, contact email) and registers the training room.
   * **Endpoint:** POST /api/training-room/register
   * The training room account is created similarly to individual user accounts but marked with a special flag (e.g., accountType: 'trainingRoom').
2. **Shared Device Scenario – Additional Student Login:**
   * **Scenario:** In an offline learning center, a computer is shared by multiple students.
   * **Workflow:**
     + When a student starts a session on a shared computer, the training room account is used to launch the web application.
     + Upon accessing a course, the system displays a “Switch User” prompt.
     + **Prompt:** “Select the active student” – a dropdown list of registered students for that training room is shown or a simple login form is displayed for the student.
     + **Endpoint:** POST /api/training-room/switch-user
       - The active student’s credentials (or a unique identifier) are provided.
       - The backend updates the session context so that all subsequent actions (access to learning materials, course progress, etc.) are attributed to the selected student rather than the training room.
     + This workflow ensures that even though the computer is shared, each student’s learning folder and progress remain distinct.

**2.2. Course Presentation and Scheduling Workflows**

**2.2.1. Course Creation by Instructors**

1. **Instructor Portal:**
   * Instructors log in through an admin/instructor dashboard.
   * They access the “Create Course” module.
2. **Course Data Entry:**
   * Form fields include:
     + Course title, description, subject area (e.g., English, Linux, Nutrition), intended audience, and course type (live, recorded, or hybrid).
     + Schedule details: Day of week, start time, duration.
     + Modules: Instructors can define a course as a series of modules (e.g., Module 1, Module 2, etc.). For each module, additional fields like title, content outline, and supplementary material references (files, links) are entered.
3. **Backend Processing:**
   * **Endpoints:**
     + POST /api/instructor/courses – Creates a new course.
     + POST /api/instructor/course-modules – Adds modules to the course.
   * The course information is stored in the Courses table, and each module is recorded in the CourseModules table.
   * The schedule is stored using the TimeSlot entity which defines when the course is available.
4. **Presentation:**
   * The course, once created, appears on the course overview page accessible to potential learners.
   * Courses are tagged with time slots and subject categories.

**2.2.2. Course Presentation to Learners**

1. **Course Listing:**
   * **Endpoint:** GET /api/courses
   * The frontend retrieves a list of courses, with filters available for subject, schedule, and course type.
   * Courses are presented with:
     + Title, description, subject, scheduled time, and number of available seats.
     + A label indicating if the course is “Live” or “Recorded.”
2. **Detailed Course View:**
   * When a user clicks on a course, a detailed view displays:
     + Full course description.
     + Module breakdown (e.g., Module 1: Introduction, Module 2: Advanced Topics).
     + Instructor details and any associated media previews.
     + A schedule view (a calendar or timeline) that shows exactly when the live sessions occur.
3. **Training Plan Generation:**
   * Learners can view a personalized training plan.
   * **Workflow:**
     + Upon booking courses, the system assembles the learner’s schedule.
     + The schedule differentiates between live sessions and recorded content.
     + A “My Schedule” dashboard is provided, showing booked courses, timings, and available recorded sessions that can fill in any gaps.
     + The logic ensures that overlapping live sessions cannot be booked, and offers suggestions for recorded sessions if there are scheduling conflicts.

**2.2.3. Course Booking Workflow for Learners**

1. **Booking Process:**
   * **Endpoint:** POST /api/bookings
   * The learner selects a course from the course listing.
   * The system checks the learner’s existing bookings to ensure no time overlap.
   * If a conflict is detected, the UI provides a warning and suggests alternatives.
   * For courses with multiple modules across different time slots, the system verifies that the learner is booking a consistent series of modules.
2. **Booking Confirmation:**
   * Once the booking is confirmed, the learner receives an on-screen confirmation and an email notification.
   * The learner’s “My Schedule” dashboard is updated accordingly.

**2.3. Live Streaming and Recording Workflows**

**2.3.1. Initiating a Live Session**

1. **Instructor Initiation:**
   * When it’s time for a live session, the instructor uses the instructor dashboard to start the stream.
   * **Endpoint:** POST /api/instructor/stream/start
   * The system signals the streaming server (NGINX-RTMP) to open a new streaming channel.
2. **User Access:**
   * Learners who have booked the course receive a notification or see the live session available on their schedule.
   * When accessing the course, the frontend embeds a live player that connects to the streaming endpoint (e.g., rtmp://<server-ip>/live/stream).
   * A live chat component (using Socket.io) is loaded alongside the video player for real-time interaction.
3. **Recording:**
   * The streaming server automatically records the session.
   * Once the live session ends, the recorded file is stored on disk or in cloud storage.
   * **Endpoint:** An API (e.g., GET /api/stream/recording/{courseId}) ensures that only learners who booked the course can retrieve the recording.

**2.3.2. Accessing Recorded Content**

1. **Learner Dashboard:**
   * The “My Schedule” dashboard shows available recorded sessions.
   * When a learner clicks on a recorded session, the system verifies booking status and then streams the recorded file.
2. **Security:**
   * Access to recordings is gated by JWT authentication and a verification check on the booking record.

**2.4. Personal Learning Folder and Media Management Workflows**

**2.4.1. Uploading and Managing Materials (Admin/Instructor Workflow)**

1. **Media Upload:**
   * On the admin/instructor dashboard, there is an option “Upload Learning Material.”
   * **Endpoint:** POST /api/upload
   * Instructors/admins upload files (PDFs, audio files) associated with specific courses and modules.
   * Metadata is captured (file type, course ID, module ID, version number).
2. **Backend Processing:**
   * The file is stored (either on local storage or a cloud storage service).
   * A new record is inserted in the LearningMaterials table with proper associations.
3. **Versioning:**
   * If the same material is re-uploaded, the system maintains version history.

**2.4.2. Learner Access**

1. **Accessing the Learning Folder:**
   * **Endpoint:** GET /api/learning-materials
   * When a learner accesses their personal dashboard (“Learning Folder”), the system queries for all materials linked to the courses they have booked.
   * The frontend displays a list of items organized by course and module.
   * Learners can click to view/download or listen to audio files.

**2.5. Group Usage in Offline Learning Centers – Detailed Workflow**

**2.5.1. Training Room Environment**

1. **Initial Login:**
   * The training room’s primary account is used to launch the application on a shared computer.
2. **Prompt for User Switch:**
   * **Scenario Trigger:** When a course is about to start or upon explicit request (e.g., “Switch User” button).
   * **User Interface:** A modal dialog appears prompting, “Please select the active student.”
   * Options include:
     + A dropdown list populated with registered student accounts for that training room.
     + Alternatively, a simple login form allowing a student to enter their credentials.
3. **Backend Processing:**
   * **Endpoint:** POST /api/training-room/switch-user
   * The system validates the student’s credentials and updates the session context.
   * All subsequent data (course bookings, learning folder access) is now associated with the selected student.
4. **Workflow Outcome:**
   * The system logs the event for audit purposes.
   * The active user’s personal schedule and learning materials are loaded immediately.

**2.6. Administration and Reporting Workflows**

**2.6.1. Admin Panel for Course and User Management**

1. **Admin Login:**
   * Administrators log in through a dedicated admin portal.
2. **User and Course Management:**
   * **Endpoints:**
     + GET /api/admin/users – Retrieve user list.
     + GET /api/admin/courses – Retrieve courses and their booking statistics.
     + POST /api/admin/courses – Create/update course details.
   * Admins can:
     + View detailed information about each course (including scheduled modules, booked seats, and attendance records).
     + Modify course details or cancel courses if needed.
     + Manage user roles (assigning Admin, Instructor, or Learner roles).
3. **Reporting:**
   * The admin dashboard displays statistics such as:
     + Number of active users.
     + Course enrollment numbers.
     + Streaming usage (e.g., average concurrent users, peak usage).
     + Access logs for learning materials.
   * Reports can be exported as CSV or PDF.

**2.6.2. Instructor Reporting and Training Plan Management**

1. **Instructor Dashboard:**
   * Instructors can log in and view their courses.
   * They see detailed training plans:
     + A breakdown of modules, timings, and participant lists.
     + Attendance and engagement metrics (e.g., chat activity during live sessions).
2. **Training Plan Creation:**
   * Instructors have tools to define or adjust the training plan:
     + They can schedule make-up sessions, update module content, or link supplementary materials.
   * The updated plan is automatically pushed to the learners’ “My Schedule” dashboards.

**3. Technical Specifications and API Endpoints Summary**

Below is a high-level list of critical endpoints and data models:

**3.1. User Management**

* **POST /api/register** – Create a new user.
* **GET /api/confirm?token=...** – Confirm user registration.
* **POST /api/login** – Authenticate user and return JWT.

**3.2. Training Room & Group Login**

* **POST /api/training-room/register** – Register a training room.
* **POST /api/training-room/switch-user** – Switch active user on shared device.

**3.3. Course Management**

* **GET /api/courses** – Retrieve course listings.
* **POST /api/bookings** – Create a booking, including schedule conflict validation.
* **GET /api/my-schedule** – Retrieve the user’s personalized course schedule.

**3.4. Instructor and Admin Functions**

* **POST /api/instructor/courses** – Create new courses.
* **POST /api/instructor/course-modules** – Add modules to courses.
* **GET /api/admin/users**, **GET /api/admin/courses** – Administration endpoints for management and reporting.

**3.5. Live Streaming and Chat**

* **POST /api/instructor/stream/start** – Initiate live stream.
* **GET /api/stream/recording/{courseId}** – Retrieve recorded session.
* **Socket.io channels** – For live chat during streams.

**3.6. Learning Materials**

* **GET /api/learning-materials** – Retrieve materials for the logged-in user.
* **POST /api/upload** – Upload media files (admin/instructor only).

**4. Deployment and Monitoring Considerations**

* **Cloud Infrastructure:**
  + Deploy backend and frontend on a cloud service (initially AWS/Azure/Google Cloud) with auto-scaling configured.
  + Streaming server deployed on a dedicated instance with recommended resources: 4–8 vCPUs, 8–16 GB RAM.
* **Scaling Triggers:**
  + Monitor CPU (>70% for 5 minutes), memory (>80% for 5 minutes), and network bandwidth usage.
  + Set up horizontal scaling via the chosen cloud platform’s auto-scaling groups.
* **Monitoring Tools:**
  + Use open-source tools such as Prometheus and Grafana for real-time monitoring.
  + Log aggregation with an ELK-stack (Elasticsearch, Logstash, Kibana).

**5. Timeline (AI-Assisted)**

Based on modern AI-assisted code generation and automated testing, the full implementation is estimated to take approximately 10–12 days, broken down as follows:

* **Day 1:** Set up the development environment, repository, CI/CD pipeline, and begin the user registration module.
* **Day 2:** Complete user registration and login workflows; implement training room registration.
* **Day 3–4:** Develop and integrate the course management and booking logic with real-time schedule conflict detection.
* **Day 5–6:** Set up and configure the live streaming server, integrate the streaming player and live chat.
* **Day 7:** Build the personal learning folder module for PDF, audio, and media management.
* **Day 8:** Implement the additional group usage workflow for offline learning centers.
* **Day 9:** Develop the admin and instructor dashboards with full reporting capabilities.
* **Day 10–11:** Integrate all modules and perform comprehensive automated testing (unit, integration, load, and security tests).
* **Day 12:** Final deployment to the cloud and set up monitoring/logging.

**Summary**

This comprehensive and highly detailed plan for TalentHíd – Tanulás élethosszig 2.0 includes:

* Detailed workflows for standard user registration and special training room scenarios.
* Step-by-step instructions for course creation, presentation, and booking with intelligent schedule management.
* Explicit workflows for live streaming initiation, recording, and real-time chat integration.
* Detailed personal learning folder management and group usage handling on shared devices.
* Administration and reporting functions for both instructors and system administrators.
* A complete list of API endpoints, data models, and technical specifications.
* Deployment strategies, monitoring, and scaling triggers.
* An AI-optimized timeline demonstrating that the entire system can be developed within approximately 10–12 days using modern code-generation tools and automated testing.

This document should serve as a complete technical blueprint for a coding-oriented AI to generate and integrate the code for all modules, ensuring a scalable, secure, and user-friendly learning platform.