**Detailed Workflow – AGILE(scrum)**

Below is the detailed workflow for developing the **Interactive Lecture Note-Sharing Platform for Handwritten PDFs**, structured around the Agile Scrum model. Each sprint includes specific tasks, deliverables, and testing to ensure progress toward a polished, functional app.

**Pre-Development Phase (Week 1)**

**Objective**: Set up the project foundation, finalize requirements, and prepare tools.

* **Tasks**:
  1. **Requirement Gathering**:
     + Confirm features: user authentication, profiles (joining year, registration number, course, college email, calculated academic year), year-based course selection, PDF upload/viewing (with module number, faculty name, tags), OCR search, ZIP downloads (by user/faculty/module), comments, points/streaks, leaderboard, analytics, notifications.
     + Define optional features: AI tags, PDF annotations, badges, course subscriptions, text-to-speech, dark mode, progress tracker, social sharing.
     + Document in Product Backlog.
  2. **Tech Stack Setup**:
     + Install Node.js, React, SQLite, Firebase (Authentication, Storage).
     + Set up libraries: Tesseract.js (OCR), pdf.js (PDF previews), JSZip (ZIP), Chart.js (visualizations), Nodemailer (notifications), Natural (AI tags).
     + Configure development environment (e.g., VS Code, Git).
  3. **Database Schema Design**:
     + Tables:
       - users (id, name, email, password, joining\_year, reg\_number, course, college\_email, points, streak).
       - notes (id, user\_id, course, faculty, module, topic, tags, file\_url, ocr\_text).
       - comments (id, note\_id, user\_id, text, timestamp).
       - subscriptions (user\_id, course\_id).
     + Initialize SQLite database.
  4. **UI Wireframes**:
     + Design layouts for signup/login, profile page, course selection, note upload/view, ZIP download, leaderboard, analytics.
     + Use Bootstrap for responsive design.
  5. **Team Setup**:
     + Assign roles (e.g., frontend, backend, testing).
     + Set up Git repository for version control.
* **Deliverables**:
  1. Product Backlog (feature list).
  2. Setup guide (tech stack installation).
  3. Initial database schema.
  4. UI wireframes.
* **Testing**: Verify tool installations and database connectivity.

**Sprint 1 (Weeks 2-3): Authentication and User Profiles**

**Objective**: Build core user management and profile display.

* **Tasks**:
  1. Implement Firebase Authentication for signup/login.
  2. Create signup form to collect name, email, password, joining year, registration number, course, college email.
  3. Develop user profile page to display details and calculate academic year (2025 - joining\_year + 1).
  4. Create backend API to retrieve profile data.
  5. Design responsive UI with Bootstrap for signup, login, and profile pages.
* **Deliverables**:
  1. Functional signup/login system.
  2. User profile page (UserProfile.js from previous artifact).
  3. Backend API for profile retrieval (new artifact below).
* **Testing**:
  1. Test signup/login with valid/invalid inputs.
  2. Verify profile displays correct details and academic year (e.g., joining 2023 = 3rd year in 2025).
  3. Test responsiveness on mobile browsers.
* **Artifact**: Backend API for user profile retrieval.

userProfile.js

javascript

Edit in files•Show inline

**Sprint 2 (Weeks 4-5): Course Selection and PDF Upload**

**Objective**: Enable course selection and note uploads with metadata.

* **Tasks**:
  1. Implement year-based course selection UI (React <select> for year, course).
  2. Create PDF upload form with fields for course, faculty, module number, topic, tags.
  3. Integrate Firebase Storage for PDF uploads and Tesseract.js for OCR.
  4. Store metadata and OCR text in SQLite (see uploadNote.js from previous artifact).
  5. Add AI tag suggestions using Natural (e.g., “bubble sort” suggests “Sorting”).
* **Deliverables**:
  1. Course selection UI.
  2. PDF upload functionality (NoteUpload.js from previous artifact).
  3. Backend API with OCR and tag suggestions.
* **Testing**:
  1. Test course selection filtering.
  2. Verify PDF uploads (size <10MB) and OCR text extraction.
  3. Check AI tag accuracy with sample notes.

**Sprint 3 (Weeks 6-7): PDF Viewing, Comments, and ZIP Downloads**

**Objective**: Add note viewing, collaboration, and bulk download features.

* **Tasks**:
  1. Implement PDF viewing with pdf.js for previews.
  2. Create commenting system (store in SQLite, display in React).
  3. Develop ZIP download UI and backend (filter by user, faculty, module; see ZipDownload.js and downloadNotes.js from previous artifacts).
  4. Add smart ZIP naming (e.g., “CS301\_Module2\_Sharma.zip”).
* **Deliverables**:
  1. PDF preview page.
  2. Commenting system.
  3. ZIP download functionality with filters.
* **Testing**:
  1. Test PDF previews on web/mobile.
  2. Verify comment posting and display.
  3. Test ZIP downloads for different filter combinations (e.g., Module 2 by Prof. Sharma).

**Sprint 4 (Weeks 8-9): Gamification and Analytics**

**Objective**: Add points/streaks, leaderboard, and usage analytics.

* **Tasks**:
  1. Implement points system (10 points per upload, +5 for streaks) and streak tracking in SQLite.
  2. Create leaderboard UI with Chart.js (see Leaderboard.js from previous artifact).
  3. Develop analytics dashboard with charts for downloads by course, faculty, module.
  4. Add cron job for daily streak resets.
* **Deliverables**:
  1. Points/streak system.
  2. Leaderboard with visualization.
  3. Analytics dashboard with charts (e.g., pie chart for module downloads).
* **Testing**:
  1. Verify points/streaks update correctly (e.g., 3-day streak = 35 points).
  2. Test leaderboard accuracy and chart rendering.
  3. Check analytics charts reflect download data.

**Sprint 5 (Weeks 10-11): Notifications, Tweaks, and Polish**

**Objective**: Add notifications, implement tweaks, and polish the app.

* **Tasks**:
  1. Implement email notifications (Nodemailer) for new notes, comments, leaderboard updates.
  2. Add course subscription for automatic notifications (store in SQLite).
  3. Implement tweaks:
     + Text-to-speech for OCR text (Web Speech API).
     + Dark mode toggle (Bootstrap or CSS variables).
     + Progress tracker for exam prep (e.g., “80% of CS301 modules downloaded”).
     + Profile contribution stats (uploads, comments).
     + Social sharing for note links (Web Share API).
  4. Optimize UI for responsiveness and performance.
* **Deliverables**:
  1. Notification system.
  2. Implemented tweaks (text-to-speech, dark mode, progress tracker, profile stats, sharing).
  3. Polished, responsive UI.
* **Testing**:
  1. Test notifications for accuracy and delivery.
  2. Verify tweak functionality (e.g., dark mode toggle, progress tracker).
  3. Test responsiveness on multiple devices (desktop, mobile).

**Sprint 6 (Week 12): Final Testing, Documentation, and Presentation**

**Objective**: Finalize the app, document, and prepare for evaluation.

* **Tasks**:
  1. Conduct end-to-end testing (signup, profiles, uploads, search, ZIP downloads, comments, gamification, analytics).
  2. Test with classmates for usability (80% should navigate without assistance).
  3. Write documentation (setup, API details, OCR challenges, gamification logic).
  4. Create user manual (how to upload, search, download ZIPs, view profiles).
  5. Record 2-3 minute demo video showing all features.
  6. Prepare presentation highlighting profiles, ZIP downloads, OCR, and gamification.
* **Deliverables**:
  1. Fully tested web app.
  2. Source code with documentation.
  3. User manual.
  4. Test report (functionality, usability, OCR accuracy).
  5. Demo video and presentation.
* **Testing**:
  1. Verify all features work as expected.
  2. Confirm ZIP downloads include correct notes.
  3. Ensure profile year calculation is accurate (e.g., joining 2023 = 3rd year in 2025).