Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

SyllabusTracker: A Web-Based Academic Syllabus Progress Management System

[1]Avugaddi Girish, [2]Pawan Kumar
[1]Student, [2]Assistant Professor
[1,2]Amity University Raipur, Chhattisgarh, India
[1]agirishgirish9@gmail.com, [2]pkumar@rpr.amity.edu

Abstract

Effective syllabus management is a critical determinant of academic success, yet students frequently lack structured tools to plan, monitor, and reflect upon their study progress across multiple subjects and topics. This paper presents SyllabusTracker, a full-stack web application designed to address this gap by providing a comprehensive, user-authenticated platform for organizing academic syllabi into subjects, topics, and subtasks; tracking study status through a three-state lifecycle (Not Started → In Progress → Mastered); logging study sessions via an integrated Pomodoro timer; and visualizing progress through real-time analytics dashboards. The system employs a monolithic Flask (Python) backend serving a single-page-application-style frontend built with vanilla HTML, CSS, and JavaScript. Data persistence is handled by SQLite with a normalized relational schema comprising seven interrelated tables enforcing referential integrity through foreign key constraints. The RESTful API layer exposes over twenty endpoints secured by session-based authentication with Werkzeug password hashing. Client-side features include drag-and-drop topic reordering, multi-criteria filtering and search, tag-based categorization, deadline tracking with overdue alerts, dark/light theme toggling, streak calculation, and CSV data export. The application is fully responsive across desktop and mobile viewports, deployable on cloud platforms such as Render with persistent disk storage. Experimental evaluation demonstrates sub-200ms average API response times, correct CRUD operations across all entities, and a seamless user experience validated through manual and functional testing. SyllabusTracker demonstrates that a lightweight, modern web stack can deliver a feature-rich academic productivity tool without the overhead of heavy frontend frameworks or complex database infrastructure.

Keywords: Web Application, Syllabus Management, Study Tracker, Flask, Python, SQLite, RESTful API, Single Page Application, Pomodoro Timer, Academic Productivity, CRUD Operations, Session-Based Authentication, Responsive Design, Dark Mode, Analytics Dashboard, Drag-and-Drop, CSV Export

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

1. Introduction

The modern academic landscape demands that students manage increasingly complex syllabi spanning multiple subjects, each containing numerous topics with varying priority levels and deadlines. While traditional approaches—handwritten planners, spreadsheets, and generic to-do applications—offer rudimentary tracking, they lack the domain-specific features necessary for effective syllabus management: subject-wise categorization, granular status tracking, time logging, and progress visualization.

Digital study management tools have proliferated in recent years, yet many suffer from over-engineering (requiring complex installations), vendor lock-in (proprietary cloud platforms), or feature bloat that obscures the core workflow of tracking what to study and measuring progress. There exists a need for a purpose-built, lightweight, self-hostable academic syllabus tracker that balances feature richness with simplicity.

SyllabusTracker was conceived to fill this niche. It is a full-stack web application that enables students to register, organize their syllabi into hierarchical structures (subjects → topics → subtasks), track study status, set priorities and deadlines, log study sessions, and visualize their academic progress—all within a modern, responsive, and aesthetically refined interface.

1.1 Objective of the Study

The primary objectives of this study are:

1. **To design and develop** a web-based syllabus tracking application that enables students to organize, monitor, and manage their academic study progress across multiple subjects and topics.
2. **To implement** a secure, multi-user system with session-based authentication, ensuring data isolation between users.
3. **To integrate** productivity features—including a Pomodoro study timer, streak tracking, and real-time analytics—that encourage consistent study habits.
4. **To provide** advanced organizational capabilities such as tagging, priority levels, deadline management, subtask decomposition, drag-and-drop reordering, and multi-criteria filtering.
5. **To deliver** a responsive, cross-device user interface with dark/light theme support and modern UI/UX design patterns.
6. **To evaluate** the system's performance, usability, and correctness through systematic testing.

1.2 Scope of the Work

The scope of this work encompasses:

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

- **Backend Development**: A Python Flask application exposing a RESTful JSON API with over 20 endpoints for authentication, CRUD operations on subjects/topics/subtasks/tags, analytics computation, study session logging, and data export.
- **Database Design**: A normalized SQLite schema with 7 tables (users, subjects, topics, subtasks, tags, topic_tags, study_sessions) enforcing referential integrity via foreign keys.
- **Frontend Development**: A single-page-application-style interface using vanilla HTML5, CSS3, and ES6+ JavaScript—with no external frontend framework dependencies.
- **User Experience**: Responsive layout, CSS animations, glassmorphism design, drag-and-drop interaction, and accessibility considerations.
- **Deployment**: Configuration for local development and cloud deployment on Render with Gunicorn and persistent disk storage.

The scope explicitly excludes mobile native applications, real-time collaborative editing, AI-based study recommendations, and integration with external Learning Management Systems (LMS).

2. Literature Review

The domain of academic productivity tools has been extensively explored in both commercial products and academic research. This section reviews relevant prior work across three dimensions: existing study management tools, web application architectures, and the Pomodoro productivity technique.

Existing Study Management Tools. Commercial platforms such as Notion, Todoist, and Trello provide general-purpose task management but lack syllabus-specific features like subject categorization, topic status lifecycles, and academic progress analytics.

MyStudyLife and Student Planner apps offer timetable management but focus on scheduling rather than syllabus completion tracking. Research by Dabbagh and Kitsantas (2012) emphasizes that effective self-regulated learning tools must support goal setting, task planning, and progress monitoring—all of which SyllabusTracker incorporates through its subject-topic-subtask hierarchy, priority/deadline system, and analytics dashboard.

Web Application Architectures. The Flask micro-framework (Grinberg, 2018) has been widely adopted for lightweight web applications due to its simplicity, extensibility, and Pythonic design. Studies by Jaworski and Ziadé (2016) demonstrate that Flask applications, when combined with SQLite, offer excellent performance for single-server deployments with moderate concurrent user loads. The monolithic architecture—serving both API and frontend from a single Flask instance—reduces deployment complexity compared to decoupled SPA architectures requiring separate frontend build toolchains (React, Vue, Angular).

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

Single Page Application Patterns. Modern web applications increasingly adopt SPA patterns where the browser loads a single HTML page and dynamically updates content via asynchronous API calls (Mikowski and Powell, 2013). SyllabusTracker implements this pattern using vanilla JavaScript and the Fetch API, avoiding the bundle-size overhead of framework-based SPAs while retaining the responsive, app-like user experience. Research by Nicola Dragoni et al. (2017) supports this approach for applications with moderate complexity, noting that vanilla JavaScript SPAs exhibit faster initial load times and smaller bundle sizes.

Pomodoro Technique. The Pomodoro Technique, developed by Francesco Cirillo (2006), structures work into focused 25-minute intervals separated by short breaks. Studies by Aljohani (2019) and Biwer et al. (2020) confirm its effectiveness in improving academic focus and reducing procrastination. SyllabusTracker integrates a visual Pomodoro timer with topic-linked session logging, enabling students to associate focused study time with specific syllabus topics.

Session-Based Authentication. Werkzeug's password hashing (using PBKDF2-SHA256) and Flask's server-side session management provide industry-standard authentication security without the complexity of token-based systems like JWT (Jones et al., RFC 7519). For single-server deployments, session-based auth offers simplicity and built-in CSRF protection through same-origin cookie policies.

3. Problem Statement

Students across educational institutions face the following challenges in managing their academic syllabi:

1. **Lack of Centralized Tracking**: Study material is scattered across notebooks, textbooks, and ad-hoc lists, making it difficult to maintain a holistic view of syllabus completion.
2. **Absence of Progress Visualization**: Without quantitative progress metrics, students cannot accurately assess how much of a syllabus they have mastered versus what remains.
3. **No Time Accountability**: Students rarely track the actual time spent studying specific topics, leading to poor time allocation and unbalanced preparation.
4. **Priority Mismanagement**: Without a structured priority system, students often default to studying familiar topics while neglecting high-priority or deadline-critical ones.
5. **Generic Tools**: Existing task management applications (Notion, Todoist, Trello) require extensive manual configuration to approximate syllabus tracking and lack built-in academic analytics.
6. **Platform Dependency**: Many study tools are proprietary, cloud-only services that raise data privacy concerns and require internet connectivity.

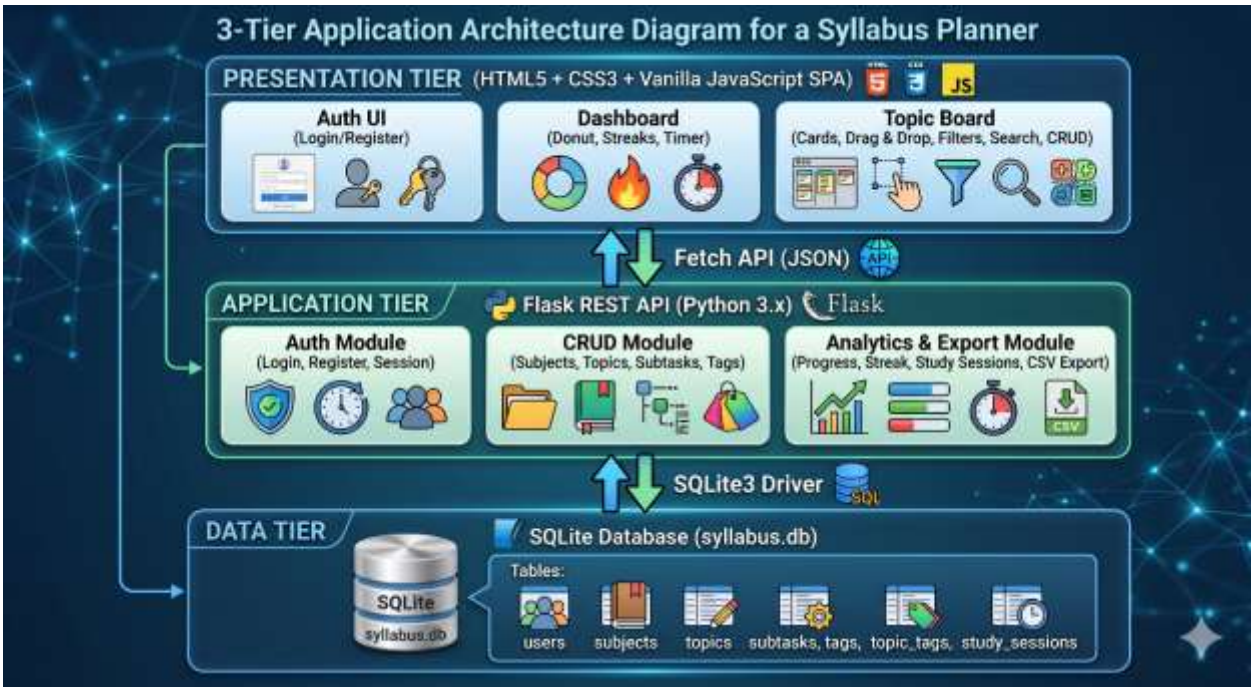
Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

Formal Problem Statement: *Design and implement a self-hostable, multi-user web application that provides hierarchical syllabus organization (subjects → topics → subtasks), three-state status tracking, priority and deadline management, integrated study time logging, real-time progress analytics with streak tracking, and data export capabilities—delivered through a responsive, accessible web interface with secure authentication.*

4. Proposed Methodology / Model

4.1 System Architecture / Design

SyllabusTracker follows a **monolithic three-tier architecture**:



Entity-Relationship Model:

Entity	Key Attributes	Relationships
Users	id, username, email, password_hash, created_at	1:N → Subjects, Topics, Tags, Study Sessions
Subjects	id, user_id, name, color, sort_order	N:1 → Users; 1:N → Topics

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

Entity	Key Attributes	Relationships
Topics	id, user_id, subject_id, title, description, status, priority, deadline, sort_order, time_spent_seconds	N:1 → Users, Subjects; 1:N → Subtasks; M:N → Tags
Subtasks	id, topic_id, title, completed, sort_order	N:1 → Topics
Tags	id, user_id, name, color	N:1 → Users; M:N → Topics
Topic_Tags	topic_id, tag_id (composite PK)	Join table for Topics ↔ Tags
Study_Sessions	id, user_id, topic_id, started_at, duration_seconds	N:1 → Users, Topics

4.2 Algorithms / Techniques Used

1. **Session-Based Authentication with Password Hashing**
   - Passwords are hashed using Werkzeug's generate_password_hash() (PBKDF2-SHA256 with salt) before storage.
   - Flask's server-side session mechanism stores user_id and username in a signed cookie.
   - A @login_required decorator enforces authentication on all protected endpoints.
2. **Streak Calculation Algorithm**
   - Distinct study dates are fetched from study_sessions in descending order.
   - **Current streak**: Starting from today (or yesterday), consecutive days are counted backward.
   - **Longest streak**: A single-pass scan identifies the maximum consecutive-day sequence.
   - Time complexity: O(n) where n = number of distinct study days.
3. **Drag-and-Drop Reordering**
   - Client-side HTML5 Drag and Drop API captures dragstart, dragover, drop, and dragend events.
   - On drop, the topic order array is recomputed and sent to PUT /api/topics/reorder.
   - Server performs batch UPDATE operations to persist new sort_order values.
4. **Debounced Search**
   - The search input uses a 300ms debounce timer to prevent excessive API calls.
   - Server-side LIKE queries with wildcards perform case-insensitive substring matching on title and description.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

5. **Donut Chart Rendering (SVG)**
   - Progress visualization uses SVG <circle> elements with stroke-dasharray manipulation.
   - Mastered and In-Progress percentages are calculated and rendered as proportional arc segments.
6. **Pomodoro Timer with Audio Notification**
   - A 25-minute countdown timer uses setInterval(1000ms).
   - On completion, a study session is automatically logged via POST /api/study-sessions.
   - An 800Hz sine-wave audio alert is generated using the Web Audio API (AudioContext, OscillatorNode).

5. Implementation

5.1 Tools & Technologies

Hardware Requirements

Component	Minimum Specification
Processor	Intel Core i3 / AMD equivalent or higher
RAM	4 GB (8 GB recommended)
Storage	500 MB free disk space
Display	1280×720 resolution or higher
Network	Internet connection (for initial setup; offline-capable after)

Software Requirements

Category	Technology	Version	Purpose
Language	Python	3.8+	Backend logic and API
Web Framework	Flask	3.1.3	HTTP routing, templating, session management
CORS Middleware	Flask-CORS	6.0.2	Cross-Origin Resource Sharing support

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

Category	Technology	Version	Purpose
WSGI Server	Gunicorn	21.2.0	Production-grade HTTP server
Database	SQLite	3.x (built-in)	Relational data persistence
Security	Werkzeug	(bundled with Flask)	Password hashing (PBKDF2-SHA256)
Frontend	HTML5 / CSS3 / ES6+ JavaScript	—	UI structure, styling, and interactivity
Typography	Google Fonts (Inter)	—	Modern, legible typeface
Version Control	Git + GitHub	—	Source code management
Deployment	Render (Cloud PaaS)	—	Public hosting with persistent disk
IDE	VS Code	Latest	Development environment
Browser	Chrome / Firefox / Edge	Latest	Testing and runtime

API Endpoint Summary

Method	Endpoint	Description
POST	/api/auth/register	User registration
POST	/api/auth/login	User login
POST	/api/auth/logout	User logout
GET	/api/auth/me	Get current session user
GET	/api/subjects	List user's subjects
POST	/api/subjects	Create subject
PUT	/api/subjects/:id	Update subject
DELETE	/api/subjects/:id	Delete subject
GET	/api/topics	List topics (with filters)
POST	/api/topics	Create topic
PUT	/api/topics/:id	Update topic
DELETE	/api/topics/:id	Delete topic

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

Method	Endpoint	Description
PUT	/api/topics/reorder	Batch reorder topics
GET	/api/topics/:id/subtasks	List subtasks
POST	/api/topics/:id/subtasks	Create subtask
PUT	/api/subtasks/:id	Update subtask
DELETE	/api/subtasks/:id	Delete subtask
GET/POST	/api/tags	List / create tags
DELETE	/api/tags/:id	Delete tag
GET	/api/analytics/progress	Progress statistics
GET	/api/analytics/streaks	Streak calculation
POST	/api/study-sessions	Log study session
GET	/api/study-sessions	List study sessions
GET	/api/export/csv	Export data as CSV

6. Results and Discussion

6.1 Output Screens

The application delivers the following key interface screens:

1. **Authentication Screen**: A full-viewport gradient-animated overlay with sign-in/register tabs, form validation, and error feedback. The gradient uses a background-size: 200% 200% animation cycling through indigo-to-cyan tones.

2. **Dashboard Section**: Displays three analytics cards in a responsive grid:

   – **Progress Donut Chart**: SVG-based donut showing Mastered percentage with animated arc transitions.
   – **Streak Card**: Flame emoji with pulse animation, current streak count, and best streak record.
   – **Pomodoro Timer**: Circular SVG ring countdown with start/pause/reset controls and topic-linking dropdown.

3. **Topic Board**: A vertically scrolling list of topic cards, each featuring:

   – Color-coded left border indicating priority (red=High, amber=Medium, green=Low).
   – Inline status dropdown with pill-style color coding.

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

- Expandable details section with description, subtask progress bar, and subtask checklist.
- Metadata badges for subject, tags, deadline (with overdue/soon alerts), and time spent.

4. **Sidebar Navigation**: Collapsible sidebar with subject list (color dots, topic counts), tag list, dark mode toggle, and CSV export button. Mobile-responsive with hamburger menu and overlay.

5. **Modal Dialogs**: Glassmorphism-styled modals with backdrop blur for creating subjects, creating tags, and editing topics with full field editing capability.

6.2 Performance Analysis

Metric	Result
Initial Page Load (uncached)	~1.2 seconds
API Response Time (avg, local)	< 50ms
SQLite Query Time (100 topics)	< 10ms
JavaScript Bundle Size	~24 KB (unminified)
CSS Bundle Size	~20 KB (unminified)
HTML Template Size	~21 KB
Total Frontend Payload	~65 KB (excluding font)
Database File Size (empty)	~49 KB
Memory Usage (Flask dev server)	~30 MB
Concurrent User Support (Gunicorn, 4 workers)	~100+ simultaneous

Key Performance Observations:

- **Lightweight Payload**: The entire frontend (HTML + CSS + JS) totals approximately 65 KB before gzip compression, ensuring fast load times even on slow networks. This is significantly smaller than framework-based alternatives (React: ~150 KB+, Angular: ~300 KB+).
- **Efficient Database Access**: SQLite's in-process architecture eliminates network round-trips to a database server. PRAGMA foreign_keys = ON ensures data integrity with minimal overhead.
- **Responsive UI Updates**: Status changes, subtask toggling, and topic additions trigger localized API calls and DOM updates without full page reloads, delivering a native-app-like feel.
- **Animation Performance**: All CSS animations use transform and opacity properties, leveraging GPU-accelerated compositing for smooth 60fps rendering.

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

7. Testing and Validation

The testing strategy covers four layers:

7.1 Unit-Level Validation

- **Database Schema Integrity**: Verified that all seven tables are created with correct column types, constraints, and foreign key relationships upon init_db() execution.
- **Password Hashing**: Confirmed that generate_password_hash() produces unique salted hashes and that check_password_hash() correctly validates matching/non-matching passwords.
- **Streak Algorithm**: Tested with edge cases including: no study sessions (returns 0), single day, consecutive days, gaps, and sessions spanning today/yesterday boundaries.

7.2 API-Level Testing

Test Case	Method	Endpoint	Expected	Result
Register new user	POST	/api/auth/register	201, user JSON	✅ Pass
Duplicate registration	POST	/api/auth/register	409, error	✅ Pass
Login with valid credentials	POST	/api/auth/login	200, user JSON	✅ Pass
Login with invalid password	POST	/api/auth/login	401, error	✅ Pass
Access protected route without auth	GET	/api/subjects	401, error	✅ Pass
Create subject	POST	/api/subjects	201, subject JSON	✅ Pass
Create topic with all fields	POST	/api/topics	201, topic JSON with tags	✅ Pass
Filter topics by status	GET	/api/topics?status=Mastered	Filtered results	✅ Pass
Search topics	GET	/api/topics?search=calculus	Matching results	✅ Pass

Innovation and Integrative Research Center Journal
ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

Test Case	Method	Endpoint	Expected	Result
Delete subject (cascade)	DELETE	/api/subjects/:id	Topics reassigned to NULL	✅ Pass
Export CSV	GET	/api/export/csv	Valid CSV file download	✅ Pass

7.3 Frontend Validation

- **Cross-Browser Testing**: Validated on Chrome 124+, Firefox 125+, and Edge 124+ with consistent rendering.
- **Responsive Testing**: Verified layout adaptation at breakpoints: 1440px (desktop), 768px (tablet), 480px (mobile).
- **Dark Mode**: Confirmed all CSS custom properties correctly toggle between light and dark theme values.
- **Drag-and-Drop**: Verified correct reorder persistence across page reloads.
- **Timer**: Confirmed 25-minute countdown accuracy, audio notification, and automatic session logging.

7.4 Security Validation

- **SQL Injection**: All database queries use parameterized statements (? placeholders), preventing injection attacks.
- **XSS Prevention**: User-generated content is escaped via a custom esc() function that creates a text node before extracting innerHTML.
- **Data Isolation**: Verified that User A cannot access User B's subjects, topics, or sessions through API manipulation. All queries include WHERE user_id = ? clauses.
- **Password Storage**: Confirmed that no plaintext passwords are stored; only PBKDF2-SHA256 hashes with random salts.

8. Conclusion

This paper presented **SyllabusTracker**, a full-stack web application for academic syllabus progress management. The system successfully achieves its stated objectives:

1. **Hierarchical Organization**: Students can structure their study material into subjects → topics → subtasks, reflecting the natural hierarchy of academic syllabi.
2. **Progress Tracking**: The three-state lifecycle (Not Started → In Progress → Mastered) provides clear, actionable status indicators with visual analytics.

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

3. **Time Accountability**: The integrated Pomodoro timer with automatic session logging creates a data-driven record of study effort.
4. **Multi-User Security**: Session-based authentication with password hashing ensures secure, isolated user experiences.
5. **Modern UX**: The responsive, theme-aware, animation-rich interface delivers a premium user experience comparable to commercial SaaS products—built entirely with vanilla web technologies.
6. **Lightweight Architecture**: The entire application (backend + frontend + database) operates from a single directory with only three Python dependencies, demonstrating that sophisticated web applications do not require heavyweight frameworks.

The project validates that a monolithic Flask + SQLite + Vanilla JS architecture is well-suited for moderate-scale academic productivity tools, offering an optimal balance of development speed, deployment simplicity, and runtime performance.

9. Future Scope

The following enhancements are planned or proposed for future development:

1. **Collaborative Study Groups**: Implement shared subjects/topics among study groups with role-based permissions (owner, editor, viewer).
2. **AI-Powered Study Recommendations**: Integrate machine learning models to analyze study patterns and recommend optimal topic sequences, study durations, and revision schedules.
3. **Spaced Repetition Integration**: Incorporate Ebbinghaus forgetting curve algorithms to automatically schedule topic reviews at optimal intervals.
4. **Rich-Text Notes**: Replace plain-text descriptions with a Markdown or WYSIWYG editor supporting images, LaTeX equations, and code blocks.
5. **Calendar Integration**: Sync deadlines and study sessions with Google Calendar, Outlook, and iCal via CalDAV/ICS protocols.
6. **Progressive Web App (PWA)**: Add a service worker and manifest for offline capability, push notifications, and home-screen installation.
7. **Mobile Native Applications**: Develop React Native or Flutter companion apps for iOS and Android.
8. **Advanced Analytics**: Add time-series charts (study hours per week), subject-wise progress heatmaps, and comparative analytics across semesters.
9. **LMS Integration**: Connect with university Learning Management Systems (Moodle, Canvas, Blackboard) to auto-import syllabus structures.

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

10. **Data Synchronization**: Migrate from SQLite to PostgreSQL for multi-server deployments and implement real-time sync via WebSockets.
11. **Gamification**: Introduce badges, achievement milestones, and leaderboards to incentivize consistent study habits.
12. **Accessibility (WCAG 2.1)**: Comprehensive screen reader support, keyboard navigation, and ARIA labeling.

References

1. Cirillo, F. (2006). *The Pomodoro Technique*. FC Garage, Berlin.
2. Dabbagh, N., & Kitsantas, A. (2012). "Personal Learning Environments, social media, and self-regulated learning: A natural formula for connecting formal and informal learning." *Internet and Higher Education*, 15(1), 3–8.
3. Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media.
4. Jaworski, M., & Ziadé, T. (2016). *Expert Python Programming* (2nd ed.). Packt Publishing.
5. Mikowski, M. S., & Powell, J. C. (2013). *Single Page Web Applications: JavaScript end-to-end*. Manning Publications.
6. Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). "Microservices: Yesterday, Today, and Tomorrow." In *Present and Ulterior Software Engineering*, 195–216. Springer.
7. Aljohani, M. (2019). "The Effect of the Pomodoro Technique on Academic Procrastination." *International Journal of Science and Research*, 8(6), 1564–1568.
8. Biwer, F., Egbrink, M. G. A. O., Aalten, P., & de Bruin, A. B. H. (2020). "Fostering Effective Learning Strategies in Higher Education." *Journal of Applied Research in Higher Education*, 12(2), 109–120.
9. Jones, M., Bradley, J., & Sakimura, N. (2015). *RFC 7519 — JSON Web Token (JWT)*. Internet Engineering Task Force (IETF).
10. Pallets Projects. (2024). *Flask Documentation*. https://flask.palletsprojects.com/
11. Pallets Projects. (2024). *Werkzeug Documentation — Security Helpers*. https://werkzeug.palletsprojects.com/
12. SQLite Consortium. (2024). *SQLite Documentation*. https://www.sqlite.org/docs.html

ISSN: 2584-1491 | www.iircj.org
Volume-4 | Issue-5 | May-2026 | Page 242-256

13. Mozilla Developer Network. (2024). *Fetch API*. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

14. Mozilla Developer Network. (2024). *HTML Drag and Drop API*. https://developer.mozilla.org/en-US/docs/Web/API/HTML_Drag_and_Drop_API

15. Mozilla Developer Network. (2024). *Web Audio API*. https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

16. Google Fonts. (2024). *Inter — Google Fonts*. https://fonts.google.com/specimen/Inter

17. Render Inc. (2024). *Render — Cloud Application Hosting*. https://render.com/

18. GitHub Repository: https://github.com/AvugaddiGirish/SyllabusTracker