Project Report

Student Name: Manshvi Sharma	**UID:** 23BDA70088
Branch: CSE-BDA	**Section/Group:** 23AIT - KRG -2A
Semester: 5th	**Date of Performance:** 20/08/25
Subject Name: Full Stack	**Subject Code:** 23CSP-339

Project Name

ReviseHub

A comprehensive web-based application designed for students and learners to manage spaced repetition revisions and daily study tasks efficiently. It combines evidence-based learning techniques with intuitive task tracking to enhance retention and productivity.

Reference Website Link / Working Demo

- **Live Demo**: https://revise-hub-ten.vercel.app/
  (Deployed on Vercel with Firebase integration for real-time data syncing. Users can sign up/login, add subjects/topics/tasks, track progress, and view analytics.)

Project Description

ReviseHub is a full-stack React application that empowers users to optimize their study routines through spaced repetition scheduling and flexible task management. Key features include:

- **User Authentication**: Secure email/password login and registration via Firebase Auth, with profile management (e.g., display name updates and password changes).
- **Spaced Revision System**: Users create topics with subtopics/problems, assign initial study dates, and generate automated revision schedules (e.g., days 1, 3, 7, 15, 30 post-initial study). Optional long-term reviews (45 days after final revision) ensure deeper retention.

- **Study Task (Todo) Management**: Add one-off or recurring tasks with due dates (today, tomorrow, specific, or undated "recommended" items). Track completion with real-time updates.
- **Subject Organization**: Separate subjects for revisions and tasks, with filtering and editing capabilities.
- **Dashboard Views**: Tabbed interfaces for revisions (Dashboard, Pending, Missed, Done, Calendar) and tasks (Today/Overdue, Tomorrow, Upcoming, Recommended, Completed, Calendar). Visual cards display progress, due dates, and subtopics.
- **Advanced Tools**:
  - **Calendar Integration**: Month-view calendar highlighting due items with modals for details.
  - **Recycle Bin**: Soft-delete with 60-day recovery window before permanent deletion.
  - **Pomodoro-Style Timer**: Built-in countdown timer for focused study sessions.
  - **Analytics Dashboard**: Reports on completion rates, stage breakdowns, daily/weekly trends, and subject performance using charts and tables.
- **UI/UX Enhancements**: Dark/light mode toggle, responsive design, modals for forms/edits, and error handling with exponential backoff for API retries.
- **Backend**: Firebase Firestore for real-time data storage (users, topics, subjects) and batch operations for efficiency.

The app promotes active recall and spaced practice, proven to improve long-term memory retention by up to 200% compared to cramming.

Problem Statement

In today's fast-paced academic environment, students often face challenges in retaining complex information over time and managing unstructured study loads. Traditional note-taking or linear review methods lead to forgetting curves (as per Ebbinghaus's research), where up to 70% of learned material is lost within 24 hours without reinforcement. Additionally, juggling deadlines, assignments, and self-study tasks results in procrastination, missed deadlines, and burnout—exacerbated by the lack of integrated tools for both scheduled revisions and ad-hoc todos.
Existing solutions like Anki (revision-focused) or Todoist (task-focused) are siloed, lacking seamless integration for holistic study planning. Learners need a unified, intuitive platform that automates spaced repetition, visualizes progress across calendars and analytics, and supports customizable workflows without

overwhelming complexity. ReviseHub addresses this by providing a student-centric tool that reduces cognitive load, boosts adherence to evidence-based learning, and delivers actionable insights to refine study habits.

High Level Design

Architecture Overview

ReviseHub follows a client-side single-page application (SPA) architecture with real-time backend synchronization:

- **Frontend (React 18+)**:
    - **State Management**: Hooks (useState, useEffect, useMemo, useCallback) for local state; no external libraries like Redux to keep it lightweight.
    - **Components**: Modular structure with reusable UI elements (e.g., Button, Modal, TopicCard) and section-specific views (RevisionSection, TaskSection, ReportsSection). Forms use controlled inputs for validation.
    - **Routing**: Implicit via state (no React Router; sections toggled via buttons).
    - **Styling**: Tailwind CSS for responsive, themeable UI (dark mode via class toggling). Icons from Lucide-React.
    - **Data Flow**: Real-time listeners (onSnapshot) fetch/update topics/subjects from Firestore. Batch writes for multi-document ops (e.g., marking done).
- **Backend (Firebase)**:
    - **Authentication**: Firebase Auth for user sessions (onAuthStateChanged listener).
    - **Database**: Firestore with nested collections (e.g., `/artifacts/spaced-revision/users/{userId}/revision_topics`). Queries use Firestore's real-time subscriptions.
    - **Storage/Security**: Rules implied via userId scoping; no file uploads.
    - **Error Handling**: Exponential backoff retries for async ops.

Key Modules

1. **Auth Module**: Handles login/register/logout; initializes user-specific data listeners.
2. **Data Models**:
   - **Topic**: `{id, name, type ('revision'/'task'), subjectId, schedule[], initialStudyDate, taskDueDate, subtopics[], enableLtr, deleted, createdAt}`.
   - **Subject**: `{id, name, type, createdAt}`.
   - **User Profile**: `{name}` stored in Firestore doc.
3. **Business Logic**:
   - **Scheduling**: Fixed intervals (`REVISION_SCHEDULE = [1,3,7,15,30]`) + long-term (45 days). `generateSchedule()` computes targets; status via `getStatusText()`.
   - **Operations**: Mark done (updates schedule/completion), soft-delete (timestamp), recover, edit (regenerates schedule if dates change).
   - **Analytics**: Memoized computations for counts, rates, trends (e.g., daily/weekly breakdowns using date utilities like `TODAY_MS`).
4. **UI Layers**: Header (nav/tabs), Main (conditional sections), Modals (add/edit/timer/bin/profile).

Deployment

- **Hosted on Vercel** for serverless frontend.
- **Firebase** for backend services.
- Scalable to 1000+ users via Firestore's real-time queries.

Diagram (Conceptual)

```
[User] --> [Firebase Auth] --> [Firestore DB]

           |

           v

[React App] <-- Real-time Sync --> [Topics/Subjects Collections]

           |

           v

[Views: Dashboard | Tasks | Reports] <-- Hooks/State --> [Modals/Tools]
```

Future Scope

ReviseHub lays a strong foundation for expansion, with potential enhancements to further personalize and scale the learning experience:

- **Mobile Responsiveness to Native App**: Convert to React Native for iOS/Android apps, adding offline support via Firebase's local caching and push notifications for due reminders.
- **AI-Powered Features**: Integrate ML (e.g., via TensorFlow.js) for adaptive scheduling—adjust intervals based on user performance (e.g., extend if consistently early completions). Natural language processing for auto-generating subtopics from notes.
- **Integrations**: Sync with Google Calendar/Apple Reminders for external due dates; export reports to PDF/CSV; connect to learning platforms like Khan Academy for auto-imported topics.
- **Social/Collaboration**: Multi-user shared study groups for collaborative revisions; gamification with streaks, badges, and leaderboards.
- **Advanced Analytics**: Predictive insights (e.g., "Risk of burnout in 3 days") using time-series forecasting; A/B testing for schedule variants.
- **Accessibility & Localization**: Full WCAG compliance, multi-language support, and voice-over for subtopics.
- **Monetization**: Freemium model with premium tiers for unlimited subjects or AI features.