**StudySwap Full-Stack Development Roadmap**

**Frontend:** React.js | **Backend:** Node.js + Express.js | **Database:** MongoDB | **Auth:** Firebase | **Storage:** Local → Firebase Storage

**All Major Pages (Frontend)**

1. Home / Landing Page — Overview, welcome, feature highlights
2. Register / Login — User authentication pages
3. Dashboard — Personalized user homepage (uploads, downloads, stats)
4. Upload Resource — Form to upload files + metadata
5. Browse / Search Resources — Search bar, filters, sorting, resource list
6. Resource Details — View resource info, comments, upvotes, download button
7. User Profile — User stats, uploaded files, reputation, badges
8. Leaderboard — Top contributors by reputation points
9. Study Groups — List and join groups
10. Group Chat — Real-time chat and shared notes inside groups
11. Settings — Account management, profile customization

**Database Collections & Schemas (MongoDB)**

* Users
  + \_id, username, email, passwordHash (if any), firebaseUID, reputationPoints, badges[], downloadCredits, uploadsCount, downloadsCount, createdAt, updatedAt
* Resources
  + \_id, uploaderUserId (ref Users), title, description, subject, semester, tags[], fileURL, fileType (pdf/doc/ppt/code), uploadDate, upvotesCount, downloadsCount, ratings[]
* Downloads
  + \_id, userId, resourceId, downloadDate
* Votes
  + \_id, userId, resourceId, voteType (up/down), votedAt
* Comments
  + \_id, userId, resourceId, content, parentCommentId (for threads), createdAt, updatedAt
* Groups
  + \_id, name, description, subject, semester, members[], createdAt
* GroupMessages
  + \_id, groupId, senderUserId, message, createdAt

**Detailed Phase-wise Roadmap**

**Phase 1: Backend Setup & Core Auth (Weeks 1–2)**

**Tasks:**

* Initialize Node.js + Express project
* Setup project structure with folders: routes/, controllers/, models/, middleware/
* Connect MongoDB (Mongoose)
* Setup Firebase Admin SDK for JWT token verification
* Implement middleware: authMiddleware to protect routes
* Create User schema and initialize DB on first login
* APIs:
  + POST /auth/login — Verify Firebase token, create/check user
  + GET /users/me — Return current user profile

✅ **Test:** Use Postman to login with a Firebase token and receive user info.

**Phase 2: Resources & File Upload API (Weeks 3–4)**

**Tasks:**

* Create Resource schema (title, description, subject, semester, tags[], fileURL, fileType, uploaderUserId, etc.)
* Integrate Multer for file uploads (local, later migrate to Firebase Storage)
* APIs:
  + POST /resources/upload — Upload file + metadata
  + GET /resources — Support filters, search, pagination
* Update user’s upload count, give credits

✅ **Test:** Upload and fetch resources with filter params in Postman.

**Phase 3: Download, Votes, and Reputation (Weeks 5–6)**

**Tasks:**

* Download Schema: userId, resourceId, timestamp
* Votes Schema: userId, resourceId, voteType
* APIs:
  + GET /resources/:id/download — Enforce credit check + log download
  + POST /votes — Upvote/downvote
* Update reputation system logic: Points for uploads, downloads, upvotes

✅ **Test:** Ensure credit system, download limits, and reputation points work.

**Phase 4: Comments, Profiles & Leaderboard (Weeks 7–8)**

**Tasks:**

* Comments Schema with parentCommentId (for threads)
* /comments CRUD APIs
* Extend /users/:id to show uploads, stats
* /leaderboard API to return top contributors by points

✅ **Test:** Post/reply/delete comments. View other users' stats.

**Phase 5: Study Groups & Real-Time Chat (Weeks 9–10)**

**Tasks:**

* Groups Schema: name, subject, semester, members[]
* GroupMessages Schema: groupId, senderUserId, message
* APIs:
  + POST /groups/create, POST /groups/join, GET /groups
* Setup WebSocket using Socket.io:
  + Join room by group ID
  + Broadcast and receive messages

✅ **Test:** Join a group and send/receive messages in real-time using Socket.io client (like Postman WebSocket feature).

**Phase 6: Storage Upgrade, Security, Deployment (Weeks 11–12)**

**Tasks:**

* Migrate from local → Firebase Storage or AWS S3 using respective SDK
* Input validation, rate limiting, API logging
* Add MongoDB indexes for search fields (subject, semester, title)
* Setup CORS, helmet, and secure headers
* Deploy backend to Heroku/Railway with environment variables

✅ **Test:** All routes via Postman. Final backend documentation via Swagger or Postman Collections.

**Frontend Development (Weeks 13–18)**

Once backend is fully tested:

**Tasks per Module (Using APIs Already Built):**

* Auth UI: Firebase Auth SDK + fetch /users/me
* Upload Page: Send to /resources/upload
* Browse/Search: Use /resources?query=xyz
* Resource Details: Use /resources/:id, /votes, /comments
* Download Button: Call /resources/:id/download
* Profile & Leaderboard: /users/:id, /leaderboard
* Study Groups & Chat: Connect to groups API + Socket.io

**🧪 Tools You Should Use During Backend Phase**

* **Postman / Insomnia** for testing APIs
* **Swagger** for API documentation (optional but recommended)
* **MongoDB Compass** to visualize data
* **Socket.io Client** or websocat for real-time WebSocket testing

**Summary Table of Pages & Collections**

| **Phase** | **Frontend Pages** | **Backend Collections & APIs** |
| --- | --- | --- |
| 1 | Home, Login/Register, Dashboard | Users, Auth endpoints |
| 2 | Upload Resource, Browse/Search | Resources, File upload APIs |
| 3 | Resource Details, Upvote UI, Downloads | Downloads, Votes APIs |
| 4 | Comments Section, User Profile, Leaderboard | Comments, User Profile, Leaderboard APIs |
| 5 | Study Groups List, Group Chat | Groups, GroupMessages, WebSocket APIs |
| 6 | UI polish, Notifications | File Storage upgrade, DB optimizations |

**Project Overview:**  
StudySwap is a full-featured web platform designed to empower students by enabling them to upload, share, browse, and download academic study resources such as PDFs, notes, presentations, and code snippets. The platform incorporates reputation-based incentives, download credit systems, real-time study group chats, and user profiles with leaderboards to foster a collaborative and engaging learning community.

**Key Features:**

* User registration and authentication via Firebase Authentication
* Resource upload with metadata (subject, semester, tags) and secure file storage
* Advanced resource browsing and search with filters and pagination
* Download system regulated by user credits and reputation points
* Upvoting, commenting (threaded), and rating of resources
* User profiles displaying uploads, downloads, reputation, and badges
* Leaderboard showcasing top contributors
* Study groups with real-time chat and group-specific resource sharing
* Responsive and polished UI/UX with smooth navigation

**Tech Stack:**

* **Frontend:** React.js for dynamic, component-driven UI
* **Backend:** Node.js with Express.js REST APIs
* **Database:** MongoDB for flexible document storage of users, resources, comments, votes, and groups
* **Authentication:** Firebase Authentication for secure user sign-in and token management
* **File Storage:** Local storage initially, planned migration to Firebase Storage or AWS S3 for scalability
* **Real-Time Communication:** WebSockets (e.g., Socket.io) for group chat functionality
* **Deployment:** Cloud platforms such as Heroku, Railway, or Vercel