Rebuild SRS with changes

Certainly! Here is your revised **Software Requirements Specification (SRS)** for the StreamX Online Video Streaming Platform, following modern industry standards, integrating your new “Enhanced MongoDB Atlas Database Design” and a clear, actionable implementation plan.

---

1. Introduction

1.1 Purpose

This document outlines the complete software requirements and technical framework for StreamX, a scalable online video streaming platform utilizing the MERN (MongoDB, Express.js, React.js, Node.js) stack.

1.2 Scope

StreamX enables registered users to manage profiles, upload, stream, and interact with video content. It features administrative moderation, analytics, monetization opportunities, and an extensible, microservices-based architecture deployed on cloud infrastructure (free-tier in MVP).

1.3 Definitions, Acronyms, and Abbreviations

- **MERN**: MongoDB, Express.js, React.js, Node.js
- **JWT**: JSON Web Token
- **MVP**: Minimum Viable Product
- **CDN**: Content Delivery Network

---

2. Overall Description

2.1 System Perspective

StreamX is a modular, horizontally scalable SaaS solution, utilizing a microservices-backend and single-page frontend architecture, with all communication over HTTPS.

2.2 User Classes and Characteristics

- **Regular Users:** Register, log in, manage profiles, upload and interact with videos.
- **Administrators:** Moderate content, manage the platform, and access analytics.

2.3 Assumptions and Constraints

- All application development leverages the MERN stack and cloud free tiers.
- Video files limited to 50MB (MVP), stored on Firebase Storage (free tier).
- MongoDB Atlas is used for database operations.
- All communication is secured by TLS 1.3.

---

3. System Features

3.1 User Management

Name	Requirement Statement	Priority
Registration	Users register with unique email/username and password	Must Have
Authentication	JWT-based authentication/refresh token handling	Must Have
User Profiles	Users manage profile info (avatar, bio, preferences)	Should Have
User Roles	Roles: user/admin	Must Have

3.2 Video Management

Name	Requirement Statement	Priority
Upload Video	Users upload videos (≤50MB, 1 per 24h) to Firebase Storage	Must Have
Video Metadata	Title, description, tags, thumbnail, and structured metadata for videos	Must Have
Video Streaming	Stream videos efficiently using public storage URLs	Must Have
Video Moderation	Administrators review/remove videos	Must Have

3.3 Community and Interaction

Name	Requirement Statement	Priority
Comments	Users may comment (limit: 50/user/24h), including replies to other comments	Must Have
Likes/Dislikes	Users can like/dislike videos (1 per user per video)	Must Have
Reporting	Users may report problematic content/comments	Should Have

3.4 Search and Discovery

Name	Requirement Statement	Priority
Search Videos	Search by video title and tags	Could Have
Video Browser	Display recent active videos on the home page	Must Have

3.5 Analytics and Reporting

Name	Requirement Statement	Priority
Basic Analytics	Show owners views/likes/comments count on their videos	Should Have
Admin Analytics	Platform stats: total users, uploads, activity	Could Have

---

4. External Interface Requirements

4.1 User Interfaces

- Responsive, mobile-first React SPA for users/admins.
- Admin dashboard for moderation and analytics.

4.2 Software Interfaces

- RESTful APIs (Express.js, Node.js)
- JWT authentication/authorization.
- Secure integration with Firebase Storage.

4.3 Communications Interfaces

- All client–server interaction over HTTPS, TLS 1.3.

---

5. Enhanced Database Design (MongoDB Atlas)

5.1 Collections and Schemas

Users Collection

See below for the full schema:

```
{
  "_id": "ObjectId",
  "username": "String (unique, req., 3-50)",
  "email": "String (unique, req., email pattern)",
  "passwordHash": "String (bcrypt)",
  "role": "String (user/admin, default: user)",
  "profile": {
    "firstName": "String (≤50, optional)",
    "lastName": "String (≤50, optional)",
    "avatar": "String (URL, optional)",
    "bio": "String (≤500, optional)"
  },
  "preferences": {
    "emailNotifications": "Boolean (default: true)",
    "publicProfile": "Boolean (default: true)"
  },
  "rateLimiting": {
    "lastUpload": "Date",
    "uploadsToday": "Number (default: 0)",
    "lastCommentTime": "Date",
    "commentsToday": "Number (default: 0)"
  },
  "isActive": "Boolean (default: true)",
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

Videos Collection

```
{
  "_id": "ObjectId",
  "userId": "ObjectId",
  "title": "String (1-100, req.)",
  "description": "String (≤2000)",
```

```
  "videoUrl": "String",
  "thumbnailUrl": "String",
  "metadata": {
    "duration": "Number (seconds)",
    "fileSize": "Number (bytes, ≤50MB)",
    "format": "String",
    "resolution": "String"
  },
  "status": "String (processing/active/removed_by_admin/removed_by_use
r)",
  "visibility": "String (public/unlisted, default: public)",
  "metrics": {
    "views": "Number (default: 0)",
    "likes": "Number (default: 0)",
    "comments": "Number (default: 0)",
    "shares": "Number (default: 0)"
  },
  "tags": "[String] (≤10)",
  "reports": [{
    "userId": "ObjectId",
    "reason": "String",
    "reportedAt": "Date",
    "status": "String (pending/reviewed/dismissed)"
  }],
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

Comments Collection

```
{
  "_id": "ObjectId",
  "videoId": "ObjectId",
  "userId": "ObjectId",
  "content": "String (1-500, req.)",
  "parentCommentId": "ObjectId (optional)",
  "isEdited": "Boolean (default: false)",
  "editedAt": "Date (optional)",
```

```
  "reports": [{
    "userId": "ObjectId",
    "reason": "String",
    "reportedAt": "Date",
    "status": "String (pending/reviewed/dismissed)"
  }],
  "isVisible": "Boolean (default: true)",
  "createdAt": "Date",
  "updatedAt": "Date"
}
```

Likes Collection

```
{
  "_id": "ObjectId",
  "videoId": "ObjectId",
  "userId": "ObjectId",
  "type": "String (like/dislike, default: like)",
  "createdAt": "Date"
}
```

Sessions Collection

```
{
  "_id": "ObjectId",
  "userId": "ObjectId",
  "refreshToken": "String (hashed)",
  "deviceInfo": {
    "userAgent": "String",
    "ipAddress": "String",
    "deviceType": "String (mobile/desktop/tablet)"
  },
  "isActive": "Boolean (default: true)",
  "expiresAt": "Date",
  "createdAt": "Date"
}
```

Analytics Collection

```
{
  "_id": "ObjectId",
  "date": "Date (daily aggregation)",
  "metrics": {
    "totalUsers": "Number",
    "activeUsers": "Number",
    "totalVideos": "Number",
    "totalViews": "Number",
    "totalComments": "Number",
    "totalLikes": "Number",
    "newRegistrations": "Number",
    "newUploads": "Number"
  },
  "createdAt": "Date"
}
```

6. Security Requirements

- All data in transit encrypted with HTTPS/TLS 1.3.
- User passwords stored with bcrypt (≥12 salt rounds).
- Admin APIs require role-based JWTs.
- API endpoints implement rate limiting.

7. Performance Requirements

- Support ≥500 concurrent users (on free-tier backend).
- Initial video playback buffering ≤5s.
- Standard page loads ≤3s.

8. Quality Requirements

- Target 98% uptime (beta).
- Maintainable codebase, clear documentation.
- Clean, intuitive UI and UX.

9. Implementation Plan

9.1 Three-Sprint Plan

Sprint 1: Foundation Setup (7 days)

- Project repo structure, DB setup (MongoDB Atlas), collection validation.
- Base backend (Node.js/Express), core middleware.
- User registration, login, JWT authentication; profile endpoints; rate limiting base logic.
- Unit tests for auth, API docs, and local development setup.

Sprint 2: Core Features (7 days)

- Video upload service (Node.js, Firebase Storage).
- Video streaming endpoints.
- Initial React frontend and authentication flow.
- Integrate backend and frontend.

Sprint 3: Community & Deployment (7 days)

- Comments/likes system.
- Admin dashboard and moderation features.
- APIs for reporting/content moderation.
- End-to-end testing, fixes, and deployment to cloud.
- Rate limiting completion and production environment adjustments.

9.2 Backend Structure (Sprint 1 Example)

```
streamx-backend/
├── src/
│   ├── controllers/
│   ├── models/
│   ├── middleware/
│   ├── routes/
│   ├── services/
│   ├── utils/
│   └── app.js
```

```
├── tests/
├── package.json
└── server.js
```

9.3 Success Criteria (Per Sprint)

- Endpoints return correct data and errors.
- Core flows (registration, login, video upload, stream, comment) function as specified.
- Rate limiting and authentication tested.
- All code documented and peer-reviewed.

---

10. Environment and Configuration

Required Environment Variables

```
MONGODB_URI=...
DB_NAME=streamx_db
JWT_SECRET=...
JWT_REFRESH_SECRET=...
JWT_EXPIRE=1h
JWT_REFRESH_EXPIRE=7d
FIREBASE_PROJECT_ID=...
FIREBASE_PRIVATE_KEY=...
FIREBASE_CLIENT_EMAIL=...
PORT=5000
NODE_ENV=development
UPLOAD_LIMIT_PER_DAY=1
COMMENT_LIMIT_PER_DAY=50
```

---

11. Appendices

11.1 Glossary

See §1.3.

11.2 References

- MongoDB Atlas Docs
- Firebase Storage Docs
- React, Express, Bcrypt Docs

End of Document

Let me know if you need this in another format or want additional sections (e.g., future enhancements, more detail on endpoints, etc.).