Database Documentation - Ujian Online System

Database Architecture (Firestore)

Sistem ini menggunakan **Firestore** sebagai database NoSQL untuk menyimpan data ujian, siswa, dan hasil. Database dirancang untuk scalable dan mendukung **Student App** dan **Admin Panel**.

Collections Overview

```
pppl-ede4b (Firebase Project)
│
├── users/                    # User data (students + admins)
│   ├── {userId}/
│   │   ├── role              # "student" | "admin" | "superadmin"
│   │   ├── name
│   │   ├── email
│   │   ├── nisn              # Only for students
│   │   ├── class             # Only for students
│   │   ├── createdAt
│   │   └── updatedAt
│
├── exams/                    # Exam/ujian data
│   ├── {examId}/
│   │   ├── title
│   │   ├── description
│   │   ├── subject
│   │   ├── grade             # Kelas target (4, 5, 6, etc)
│   │   ├── duration          # in minutes
│   │   ├── totalQuestions
│   │   ├── passingScore      # Minimum score to pass (0-100)
│   │   ├── isActive          # boolean
│   │   ├── createdBy         # userId of creator (admin)
│   │   ├── createdAt
│   │   ├── updatedAt
│   │   └── scheduledDate     # Optional: scheduled exam date
│
├── questions/                # Question bank (separate for reusability)
│   ├── {questionId}/
│   │   ├── examId            # Reference to exam
│   │   ├── questionText
│   │   ├── questionNumber    # Order in exam
│   │   ├── options           # Array of 4 strings
│   │   ├── correctAnswer     # Index (0-3)
│   │   ├── subject
│   │   ├── difficulty        # "easy" | "medium" | "hard"
│   │   ├── explanation       # Optional: penjelasan jawaban
│   │   └── imageUrl          # Optional: question image
│
├── examAttempts/             # Student exam submissions
```

```
│   ├── {attemptId}/
│   │   ├── examId
│   │   ├── examTitle
│   │   ├── studentId
│   │   ├── studentName
│   │   ├── studentClass
│   │   ├── answers             # Map: { questionId: answerIndex }
│   │   ├── score               # 0-100
│   │   ├── totalQuestions
│   │   ├── correctAnswers
│   │   ├── incorrectAnswers
│   │   ├── unanswered
│   │   ├── timeSpent           # in seconds
│   │   ├── isPassed            # boolean
│   │   ├── startedAt
│   │   ├── submittedAt
│   │   └── status              # "completed" | "abandoned"
│
├── analytics/                  # Aggregated analytics (for AI)
│   ├── {analyticsId}/
│   │   ├── examId
│   │   ├── totalAttempts
│   │   ├── averageScore
│   │   ├── passRate            # Percentage
│   │   ├── questionStats       # Array of { questionId, correctRate }
│   │   ├── classStats          # Stats per class
│   │   └── lastUpdated
│
└── settings/                   # System settings
    ├── config/
    │   ├── schoolName
    │   ├── schoolAddress
    │   ├── academicYear
    │   └── geminiApiKey        # For admin panel AI
```

Detailed Schema

1. Collection: `users`

Stores all user data (students and admins).

Field	Type	Required	Description
uid	string	✅	Firebase Auth UID (document ID)
role	string	✅	"student", "admin", or "superadmin"
name	string	✅	Full name
email	string	✅	Email (format: nisn@student.sdnpgs1.sch.id for students)
nisn	string	❌	NISN (only for students)

Field	Type	Required	Description
class	string	❌	Class (e.g., "4A", "5B") - only for students
phoneNumber	string	❌	Optional contact
isActive	boolean	✅	Account active status
createdAt	timestamp	✅	Account creation date
updatedAt	timestamp	✅	Last update

Example Document (Student):

```
{
  uid: "abc123",
  role: "student",
  name: "Ahmad Rizki",
  email: "1234567890@student.sdnpgs1.sch.id",
  nisn: "1234567890",
  class: "4A",
  isActive: true,
  createdAt: Timestamp,
  updatedAt: Timestamp
}
```

Example Document (Admin):

```
{
  uid: "xyz789",
  role: "admin",
  name: "Ibu Siti Guru",
  email: "siti.guru@sdnpgs1.sch.id",
  isActive: true,
  createdAt: Timestamp,
  updatedAt: Timestamp
}
```

---

2. Collection: `exams`

Stores exam/test metadata.

Field	Type	Required	Description
id	string	✅	Auto-generated document ID
title	string	✅	Exam title
description	string	✅	Exam description

Field	Type	Required	Description
subject	string	✅	Subject (Matematika, IPA, etc)
grade	number	✅	Target grade/class (4, 5, 6)
duration	number	✅	Duration in minutes
totalQuestions	number	✅	Total number of questions
passingScore	number	✅	Minimum score to pass (0-100)
isActive	boolean	✅	Is exam currently active
createdBy	string	✅	Admin user ID
createdAt	timestamp	✅	Creation date
updatedAt	timestamp	✅	Last update
scheduledDate	timestamp	❌	Optional scheduled date

Example Document:

```
{
  id: "exam_001",
  title: "Ujian Matematika - Perkalian dan Pembagian",
  description: "Ujian tengah semester tentang perkalian dan pembagian",
  subject: "Matematika",
  grade: 4,
  duration: 45,
  totalQuestions: 20,
  passingScore: 65,
  isActive: true,
  createdBy: "xyz789",
  createdAt: Timestamp,
  updatedAt: Timestamp,
  scheduledDate: Timestamp
}
```

---

3. Collection: `questions`

Stores individual questions. Separated from exams for reusability.

Field	Type	Required	Description
id	string	✅	Auto-generated document ID
examId	string	✅	Reference to parent exam
questionText	string	✅	Question text
questionNumber	number	✅	Order in exam (1, 2, 3...)

Field	Type	Required	Description
options	array	✅	Array of 4 answer options
correctAnswer	number	✅	Index of correct answer (0-3)
subject	string	✅	Subject category
difficulty	string	❌	"easy", "medium", "hard"
explanation	string	❌	Explanation of answer
imageUrl	string	❌	Optional image URL
createdAt	timestamp	✅	Creation date

Example Document:

```
{
  id: "q_001",
  examId: "exam_001",
  questionText: "Berapa hasil dari 7 × 8?",
  questionNumber: 1,
  options: ["54", "56", "58", "60"],
  correctAnswer: 1,
  subject: "Matematika",
  difficulty: "easy",
  explanation: "7 × 8 = 56. Cara menghitung: 7 × 8 sama dengan 7 ditambah
8 kali.",
  imageUrl: null,
  createdAt: Timestamp
}
```

---

4. Collection: `examAttempts`

Stores student exam submissions and results.

Field	Type	Required	Description
id	string	✅	Auto-generated document ID
examId	string	✅	Reference to exam
examTitle	string	✅	Exam title (denormalized)
studentId	string	✅	Student user ID
studentName	string	✅	Student name (denormalized)
studentClass	string	✅	Student class (denormalized)
answers	map	✅	Map of questionId: answerIndex

Field	Type	Required	Description
score	number	✅	Final score (0-100)
totalQuestions	number	✅	Total questions
correctAnswers	number	✅	Number of correct answers
incorrectAnswers	number	✅	Number of incorrect answers
unanswered	number	✅	Number of unanswered
timeSpent	number	✅	Time spent in seconds
isPassed	boolean	✅	Did student pass?
startedAt	timestamp	✅	When exam started
submittedAt	timestamp	✅	When exam submitted
status	string	✅	"completed" or "abandoned"

Example Document:

```
{
  id: "attempt_001",
  examId: "exam_001",
  examTitle: "Ujian Matematika - Perkalian dan Pembagian",
  studentId: "abc123",
  studentName: "Ahmad Rizki",
  studentClass: "4A",
  answers: {
    "q_001": 1,  // answered option 1
    "q_002": 2,  // answered option 2
    "q_003": 0,  // answered option 0
    // ... etc
  },
  score: 85,
  totalQuestions: 20,
  correctAnswers: 17,
  incorrectAnswers: 3,
  unanswered: 0,
  timeSpent: 1800,  // 30 minutes
  isPassed: true,
  startedAt: Timestamp,
  submittedAt: Timestamp,
  status: "completed"
}
```

---

5. Collection: `analytics`

Aggregated analytics for AI analysis.

Field	Type	Required	Description
id	string	✅	Format: "analytics_{examId}"
examId	string	✅	Reference to exam
totalAttempts	number	✅	Total exam attempts
averageScore	number	✅	Average score
passRate	number	✅	Pass percentage
questionStats	array	✅	Per-question statistics
classStats	map	✅	Stats grouped by class
lastUpdated	timestamp	✅	Last calculation time

Example Document:

```
{
  id: "analytics_exam_001",
  examId: "exam_001",
  totalAttempts: 45,
  averageScore: 78.5,
  passRate: 82.2,  // 82.2% passed
  questionStats: [
    {
      questionId: "q_001",
      questionNumber: 1,
      correctRate: 95.5,  // 95.5% answered correctly
      commonWrongAnswer: 0  // Most common wrong answer index
    },
    // ... more questions
  ],
  classStats: {
    "4A": { attempts: 20, averageScore: 82.0, passRate: 90 },
    "4B": { attempts: 25, averageScore: 75.2, passRate: 76 }
  },
  lastUpdated: Timestamp
}
```

---

6. Collection: `settings`

System-wide settings.

Document: `config`

```
{
  schoolName: "SDN Pasir Gunung Selatan 1",
  schoolAddress: "Jl. ...",
  academicYear: "2024/2025",
  geminiApiKey: "AIza...",  // For admin AI features
  maintenanceMode: false,
  updatedAt: Timestamp
}
```

Database Relationships (ER Diagram)

```
┌──────────────┐
│    users     │
│  (students)  │
└──────────────┘
       │
       │ 1:N (one student, many attempts)
       │
       ▼
┌───────────────┐
│examAttempts   │
└───────────────┘
       │
       │ N:1 (many attempts, one exam)
       │
       ▼
┌──────────────┐          ┌───────────────┐
│    exams     │ ◄────────│   questions   │
└──────────────┘   1:N    └───────────────┘
       │
       │ 1:1 (optional)
       │
       ▼
┌───────────────┐
│   analytics   │
└───────────────┘

┌──────────────┐
│    users     │
│   (admins)   │
└──────────────┘
       │
       │ 1:N (one admin, many exams)
       │
       ▼
┌──────────────┐
│    exams     │
└──────────────┘
```

Indexes Required (for Performance)

Create these composite indexes in Firestore:

Collection: `examAttempts`

1. `studentId` (Ascending) + `submittedAt` (Descending)
2. `examId` (Ascending) + `submittedAt` (Descending)
3. `studentClass` (Ascending) + `score` (Descending)

Collection: `questions`

1. `examId` (Ascending) + `questionNumber` (Ascending)

Collection: `exams`

1. `isActive` (Ascending) + `createdAt` (Descending)
2. `grade` (Ascending) + `isActive` (Ascending)

Security Considerations

1. **Denormalization**: `examTitle`, `studentName`, `studentClass` disimpan di `examAttempts` untuk faster queries tanpa joins
2. **Analytics Collection**: Pre-computed untuk menghindari heavy queries saat AI analysis
3. **Separate Questions**: Questions dalam collection terpisah supaya bisa reusable di multiple exams (future feature)

Admin Panel Integration

Admin panel akan punya akses ke:

- **Read/Write**: `exams`, `questions`, `analytics`, `settings`
- **Read Only**: `examAttempts` (tidak bisa edit hasil siswa)
- **Full Access**: `users` (manage students & admins)

AI Features di Admin Panel:

- Query `analytics` collection untuk insights
- Pass data ke Gemini API untuk recommendations
- Store API key di `settings/config`

Migration Notes

Karena ini fresh database, follow steps:

1. Create collections manually via script
2. Setup security rules
3. Create indexes
4. Seed sample data
5. Test dengan student app

6. Build admin panel

Next: Security Rules & Seeding Script