MODULE 1: DATABASE DESIGN & SECURITY

WHERE IS MY BUS? - Backend Architecture Documentation

Project: Where Is My Bus? - Smart Bus Tracking System
Developer: Arpit Anand (23BCS12710)
Module: 1 of 6 - Database Design & Security
Technology Stack: MongoDB, JWT, BCrypt, Spring Security
Status: Planning & Documentation Phase

1. OVERVIEW

Module 1 establishes the **data foundation and security infrastructure** for the entire Where Is My Bus? platform. This module handles:

- **Database Schema Design** - MongoDB collections with proper relationships
- **Authentication & Authorization** - JWT-based secure access control
- **Data Security** - Encryption, validation, and protection mechanisms
- **User Management** - Role-based access control (RBAC)

2. DATABASE TECHNOLOGY CHOICE

2.1 Why MongoDB?

Criterion	MongoDB Advantage
Flexible Schema	Easy to add new fields (e.g., new badge types) without migration
JSON-like Documents	Perfect match with React frontend (JSON responses)
Geospatial Queries	Built-in support for location-based queries (buses, stops)
Scalability	Horizontal scaling for future growth
Performance	Fast read/write for real-time tracking

2.2 Database Architecture

```
MongoDB Atlas (Cloud)
├── Production Database
├── Development Database
└── Test Database
```

3. DATABASE COLLECTIONS

3.1 USERS COLLECTION

Purpose: Store all users (passengers, drivers, admins)

```
{
  _id: ObjectId("507f1f77bcf86cd799439011"),

  // Basic Information
  name: "Arjun Kumar",
  email: "arjun@example.com",
  phone: "+919876543210",
  role: "PASSENGER", // PASSENGER, DRIVER, ADMIN
  profilePicture: "https://s3.amazonaws.com/users/profile_xyz.jpg",

  // Authentication
  passwordHash: "$2b$12$xK8j9L2m.wN3oP5qR7sT...", // BCrypt hashed
  passwordSalt: "random_salt_string",
  passwordChangedAt: ISODate("2025-10-15T08:30:00Z"),
  refreshToken: "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",

  // Account Status
  accountStatus: "ACTIVE", // ACTIVE, SUSPENDED, BANNED, DELETED
  suspensionReason: null,
  suspensionEndDate: null,
  emailVerified: true,
  phoneVerified: true,

  // Gamification (Passenger only)
  trustScore: 850,
  points: 1250,
  level: "EAGLE_EYE",
  badges: [
    {
      badgeId: "badge_scout",
      name: "SCOUT",
      unlockedAt: ISODate("2025-09-01T10:00:00Z")
    },
    {
      badgeId: "badge_eagle",
      name: "EAGLE_EYE",
      unlockedAt: ISODate("2025-10-15T14:30:00Z")
    }
  ],

  // Statistics
  reportCount: {
    submitted: 125,
    approved: 118,
    rejected: 7
  },

  // Driver-Specific Fields (only for drivers)
  driverDetails: {
    licenseNumber: "DL01AA1234567",
    licenseExpiry: ISODate("2026-12-31T00:00:00Z"),
    assignedRouteId: ObjectId("route_42"),
    assignedBusId: ObjectId("bus_789"),
    performanceRating: 4.8,
    totalTrips: 342,
    onTimePercentage: 92.5
  },

  // Location Permissions
  locationPermissions: {
    isGranted: true,
    lastLocationSync: ISODate("2025-11-06T10:30:00Z")
  },

  // Preferences
  notificationPreferences: {
    emailNotifications: true,
    pushNotifications: true,
    inAppNotifications: true,
    smsNotifications: false
  },

  privacySettings: {
    showProfilePublic: false,
    allowMessagesFromPassengers: true
  },

  // Audit Trail
  createdAt: ISODate("2025-08-10T12:00:00Z"),
  updatedAt: ISODate("2025-11-06T10:30:00Z"),
  lastLoginDate: ISODate("2025-11-06T10:15:00Z"),
  lastActivityDate: ISODate("2025-11-06T10:30:00Z"),
  loginHistory: [
    {
      timestamp: ISODate("2025-11-06T10:15:00Z"),
      ipAddress: "103.24.56.78",
      device: "Android Mobile",
      location: "Chandigarh, India"
    }
  ]
}
```

Indexes:

```
db.users.createIndex({ email: 1 }, { unique: true })
db.users.createIndex({ phone: 1 }, { unique: true })
db.users.createIndex({ role: 1 })
db.users.createIndex({ trustScore: -1 }) // For leaderboard
db.users.createIndex({ "driverDetails.licenseNumber": 1 }, { sparse: true })
```

Validation Rules:

Field	Constraint	Validation
email	Required, Unique	Email regex pattern
phone	Required, Unique	Indian phone format
role	Required	Enum: PASSENGER, DRIVER, ADMIN
passwordHash	Required	BCrypt hash (60 chars)
trustScore	Optional	Number >= 0, default 0

3.2 BUSES COLLECTION

Purpose: Store all bus fleet information

```
{
  _id: ObjectId("bus_789"),

  // Bus Identification
  busNumber: "HR26Q4321",
  registrationNumber: "HR-26-Q-4321",

  // Bus Details
  busType: "AC_DELUXE", // AC_DELUXE, NON_AC, SLEEPER
  capacity: {
    seating: 45,
    standing: 20,
    total: 65
  },
  manufacturer: "Ashok Leyland",
  model: "Viking",
  yearOfManufacture: 2022,

  // Current Status
  status: "ACTIVE", // ACTIVE, MAINTENANCE, RETIRED, OFFLINE
  operationalStatus: "IN_SERVICE", // IN_SERVICE, ON_ROUTE, IDLE, BREAKDOWN

  // Assignment
  assignedRouteId: ObjectId("route_42"),
  assignedDriverId: ObjectId("driver_user_123"),

  // Current Location (Real-time)
  currentLocation: {
    type: "Point",
    coordinates: [77.1025, 28.7041] // [longitude, latitude]
  },
  locationUpdatedAt: ISODate("2025-11-14T13:45:00Z"),
  speed: 35, // km/h
  heading: 180, // degrees (0-360)

  // Route Progress
  currentStopId: ObjectId("stop_main_street"),
  nextStopId: ObjectId("stop_sector_17"),
  stopsCompleted: 5,
  totalStops: 12,

  // Features
  features: ["GPS", "AC", "WiFi", "USB_CHARGING"],

  // Maintenance
  lastMaintenanceDate: ISODate("2025-10-01T00:00:00Z"),
  nextMaintenanceDate: ISODate("2025-12-01T00:00:00Z"),
  maintenanceHistory: [
    {
      date: ISODate("2025-10-01T00:00:00Z"),
      type: "ROUTINE",
      cost: 8500,
      description: "Oil change, tire rotation, brake check"
    }
  ],

  // Timestamps
  createdAt: ISODate("2022-08-15T00:00:00Z"),
  updatedAt: ISODate("2025-11-14T13:45:00Z")
}
```

Indexes:

```
db.buses.createIndex({ busNumber: 1 }, { unique: true })
db.buses.createIndex({ registrationNumber: 1 }, { unique: true })
db.buses.createIndex({ status: 1 })
db.buses.createIndex({ assignedRouteId: 1 })
db.buses.createIndex({ currentLocation: "2dsphere" }) // Geospatial
```

3.3 ROUTES COLLECTION

Purpose: Store all bus routes with stops

```
{
  _id: ObjectId("route_42"),

  // Route Identification
  routeNumber: "42",
  routeName: "Chandigarh - Panchkula Express",

  // Route Details
  origin: "ISBT Sector 43, Chandigarh",
  destination: "Panchkula Bus Stand",
  distance: 18.5, // km
  estimatedDuration: 45, // minutes

  // Stops (Ordered sequence)
  stops: [
    {
      stopId: ObjectId("stop_isbt"),
      stopName: "ISBT Sector 43",
      location: {
        type: "Point",
        coordinates: [76.7933, 30.7333]
      },
      sequenceNumber: 1,
      arrivalTime: "06:00",
      departureTime: "06:05",
      waitTime: 5, // minutes
      distanceFromOrigin: 0 // km
    },
    {
      stopId: ObjectId("stop_sector17"),
      stopName: "Sector 17 Plaza",
      location: {
        type: "Point",
        coordinates: [76.7794, 30.7409]
      },
      sequenceNumber: 2,
      arrivalTime: "06:15",
      departureTime: "06:18",
      waitTime: 3,
      distanceFromOrigin: 3.2
    }
    // ... more stops
  ],

  // Operating Details
  operatingDays: ["MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY", "SATURDAY"],
  frequency: 15, // minutes between buses
  firstBus: "05:30",
  lastBus: "23:00",

  // Fare Structure
  fare: {
    baseFare: 10,
    perKmCharge: 2,
    maxFare: 50,
    acSurcharge: 10
  },

  // Status
  status: "ACTIVE", // ACTIVE, SUSPENDED, UNDER_REVISION

  // Statistics
  stats: {
    totalBusesAssigned: 12,
    avgPassengersPerTrip: 42,
    avgDelayMinutes: 5,
    onTimePercentage: 85
  },

  // Timestamps
  createdAt: ISODate("2023-01-15T00:00:00Z"),
  updatedAt: ISODate("2025-11-10T00:00:00Z")
}
```

Indexes:

```
db.routes.createIndex({ routeNumber: 1 }, { unique: true })
db.routes.createIndex({ status: 1 })
db.routes.createIndex({ "stops.location": "2dsphere" })
```

3.4 REPORTS COLLECTION

Purpose: Store community bus sighting reports

```
{
  _id: ObjectId("report_001"),

  // Report Identification
  reportId: "RPT_20251106_001",
  reportType: "BUS_SIGHTING", // BUS_SIGHTING, DELAY, ISSUE, FEEDBACK

  // Reporter Information
  passengerId: ObjectId("user_pass_123"),
  passengerName: "Arjun Kumar",
  passengerTrustScore: 850,
  passengerBadge: "EAGLE_EYE",

  // Bus Information
  busNumber: "HR26Q4321",
  busId: ObjectId("bus_789"),
  routeId: ObjectId("route_42"),

  // Location Data
  location: {
    type: "Point",
    coordinates: [77.1025, 28.7041] // [lon, lat]
  },
  gpsAccuracy: 15.5, // meters
  gpsTimestamp: ISODate("2025-11-06T10:30:00Z"),
  nearestStopId: ObjectId("stop_main_street"),
  stopName: "Main Street Market",

  // Photo Evidence
  photo: {
    url: "https://s3.amazonaws.com/reports/photo_abc123.jpg",
    thumbnailUrl: "https://s3.amazonaws.com/reports/thumb_abc123.jpg",
    uploadedAt: ISODate("2025-11-06T10:30:05Z"),
    size: 2048576, // bytes
    format: "image/jpeg",
    width: 1920,
    height: 1080
  },

  // Report Content
  description: "Bus arrived at Main Street Market stop, on schedule",
  reportDetails: {
    busFull: false,
    crowdLevel: "MEDIUM", // LOW, MEDIUM, HIGH
    drivingBehavior: "NORMAL", // NORMAL, RASH, SLOW
    cleanliness: "GOOD", // GOOD, AVERAGE, BAD
    acWorking: true,
    onTime: true
  },

  // Validation Status
  status: "APPROVED", // PENDING, APPROVED, REJECTED, SPAM
  validationTimestamp: ISODate("2025-11-06T10:45:00Z"),

  // Admin Review
  adminId: ObjectId("admin_user_789"),
  adminName: "System Admin",
  adminNotes: "",
  rejectionReason: null, // "WRONG_BUS", "FAKE_PHOTO", "POOR_GPS"

  // Points &amp; Trust Impact
  pointsAwarded: 15, // 10 for sighting + 5 for photo
  trustAdjustment: 1.5, // +1.5 for approved report with photo

  // Verification
  verificationStatus: "VERIFIED", // VERIFIED, UNVERIFIED, DISPUTED
  conflictingReports: [], // IDs of contradicting reports

  // Audit Trail
  submittedAt: ISODate("2025-11-06T10:30:10Z"),
  approvedAt: ISODate("2025-11-06T10:45:30Z"),
  createdAt: ISODate("2025-11-06T10:30:10Z"),
  updatedAt: ISODate("2025-11-06T10:45:30Z")
}
```

Indexes:

```
db.reports.createIndex({ reportId: 1 }, { unique: true })
db.reports.createIndex({ status: 1, submittedAt: -1 })
db.reports.createIndex({ passengerId: 1 })
db.reports.createIndex({ busNumber: 1 })
db.reports.createIndex({ location: "2dsphere" })
db.reports.createIndex({ submittedAt: -1 })
```

3.5 TRIPS COLLECTION

Purpose: Store individual bus trip records

```
{
  _id: ObjectId("trip_001"),

  // Trip Identification
  tripId: "TRIP_20251114_001",

  // Assignment
  busId: ObjectId("bus_789"),
  busNumber: "HR26Q4321",
  routeId: ObjectId("route_42"),
  driverId: ObjectId("driver_user_123"),
  driverName: "Rajesh Sharma",

  // Trip Status
  status: "COMPLETED", // SCHEDULED, IN_PROGRESS, COMPLETED, CANCELLED

  // Timing
  scheduledStartTime: ISODate("2025-11-14T06:00:00Z"),
  actualStartTime: ISODate("2025-11-14T06:02:00Z"),
  scheduledEndTime: ISODate("2025-11-14T07:00:00Z"),
  actualEndTime: ISODate("2025-11-14T07:05:00Z"),
  totalDuration: 63, // minutes

  // Route Progress
  stops: [
    {
      stopId: ObjectId("stop_isbt"),
      stopName: "ISBT Sector 43",
      sequenceNumber: 1,
      scheduledArrival: ISODate("2025-11-14T06:00:00Z"),
      actualArrival: ISODate("2025-11-14T06:02:00Z"),
      scheduledDeparture: ISODate("2025-11-14T06:05:00Z"),
      actualDeparture: ISODate("2025-11-14T06:07:00Z"),
      status: "COMPLETED",
      delay: 2, // minutes
      passengersBoarded: 12,
      passengersAlighted: 0
    }
    // ... all stops
  ],

  // Performance Metrics
  metrics: {
    totalStops: 15,
    stopsCompleted: 15,
    onTimeStops: 12,
    delayedStops: 3,
    avgDelayPerStop: 2.5, // minutes
    totalPassengers: 156,
    peakOccupancy: 62,
    avgOccupancy: 45
  },

  // Distance &amp; Fuel
  distanceCovered: 18.7, // km
  fuelConsumed: 4.2, // liters

  // Issues
  issues: [
    {
      timestamp: ISODate("2025-11-14T06:30:00Z"),
      issueType: "TRAFFIC_JAM",
      description: "Heavy traffic at Sector 17",
      delayMinutes: 8
    }
  ],

  // Timestamps
  createdAt: ISODate("2025-11-14T05:00:00Z"),
  updatedAt: ISODate("2025-11-14T07:05:30Z")
}
```

Indexes:

```
db.trips.createIndex({ tripId: 1 }, { unique: true })
db.trips.createIndex({ busId: 1, actualStartTime: -1 })
db.trips.createIndex({ driverId: 1, actualStartTime: -1 })
db.trips.createIndex({ routeId: 1 })
db.trips.createIndex({ status: 1 })
```

3.6 MESSAGES COLLECTION

Purpose: Store passenger-driver communications

```
{
  _id: ObjectId("msg_001"),

  // Message Identification
  messageId: "MSG_20251114_001",

  // Participants
  senderId: ObjectId("passenger_user_123"),
  senderName: "Arjun Kumar",
  senderRole: "PASSENGER",
  receiverId: ObjectId("driver_user_456"),
  receiverName: "Rajesh Sharma",
  receiverRole: "DRIVER",

  // Message Content
  messageType: "TEXT", // TEXT, IMAGE, LOCATION
  content: "Will the bus reach Sector 17 stop in 5 minutes?",

  // Status
  status: "DELIVERED", // SENT, DELIVERED, READ
  readAt: ISODate("2025-11-14T10:32:00Z"),

  // Context
  busId: ObjectId("bus_789"),
  routeId: ObjectId("route_42"),

  // Timestamps
  sentAt: ISODate("2025-11-14T10:30:00Z"),
  createdAt: ISODate("2025-11-14T10:30:00Z")
}
```

Indexes:

```
db.messages.createIndex({ senderId: 1, receiverId: 1, sentAt: -1 })
db.messages.createIndex({ busId: 1 })
db.messages.createIndex({ status: 1 })
```

4. AUTHENTICATION & SECURITY

4.1 JWT (JSON Web Token) Authentication

Token Structure:

```
Header.Payload.Signature
```

Access Token (Short-lived - 15 minutes):

```
{
  "header": {
    "alg": "HS256",
    "typ": "JWT"
  },
  "payload": {
    "sub": "507f1f77bcf86cd799439011", // User ID
    "email": "arjun@example.com",
    "role": "PASSENGER",
    "name": "Arjun Kumar",
    "iat": 1699000000, // Issued at
    "exp": 1699000900  // Expires in 15 min
  },
  "signature": "HMACSHA256(...)"
}
```

Refresh Token (Long-lived - 7 days):

Stored in database `users.refreshToken` field. Used to generate new access tokens without re-login.

4.2 Password Security

BCrypt Hashing:

- **Cost Factor:** 12 (2^12 iterations)
- **Salt:** Automatically generated per password
- **Hash Length:** 60 characters

Example:

```
// Plain password: "MySecurePass123"
// BCrypt hash: "$2b$12$xK8j9L2m.wN3oP5qR7sTuOQjBkYIefPmv9Z8xGqKlYzT1XrH2YUy."
```

4.3 Role-Based Access Control (RBAC)

Role	Permissions
PASSENGER	View buses, report sightings, view rewards, message driver
DRIVER	Manage trips, check-in stops, view assigned route, respond to messages
ADMIN	Full CRUD on routes/buses/users, validate reports, view analytics

4.4 API Security Headers

```
Authorization: Bearer &lt;JWT_TOKEN&gt;
Content-Type: application/json
X-API-Key: &lt;API_KEY&gt; (for external integrations)
```

5. DATA VALIDATION RULES

5.1 User Registration

Field	Validation
Name	Required, 2-50 chars, letters only
Email	Required, valid email format, unique
Phone	Required, Indian format +91XXXXXXXXXX, unique
Password	Required, min 8 chars, 1 uppercase, 1 number
Role	Required, enum: PASSENGER, DRIVER, ADMIN

5.2 Bus Report Submission

Field	Validation
Bus Number	Required, format: HR26Q4321
Location	Required, GPS accuracy < 50m
Photo	Optional, max 5MB, JPEG/PNG/WebP
Description	Optional, max 1000 chars

5.3 Route Creation

Field	Validation
Route Number	Required, alphanumeric, unique
Origin	Required, string
Destination	Required, string
Stops	Required, min 2 stops, ordered
Distance	Required, positive number

6. DATA ENCRYPTION

6.1 At Rest

- **MongoDB Encryption:** AES-256 encryption enabled
- **S3 Bucket:** Server-side encryption (SSE-S3)
- **Backup Encryption:** Encrypted backups on AWS

6.2 In Transit

- **HTTPS:** All API calls over TLS 1.3
- **WebSocket:** Secure WebSocket (WSS)

6.3 Sensitive Fields Encryption

```
// Fields to encrypt at application level:
- users.phone (encrypted with app secret)
- users.refreshToken (encrypted)
- driverDetails.licenseNumber (encrypted)
```

7. DATABASE RELATIONSHIPS

```
┌─────────┐     assigns      ┌──────┐
│  USERS  │ ──────────────────&gt;│ BUSES  │
│(Drivers)│                 │      │
└─────────┘                  └──────┘
     │                          │
     │ creates                  │ runs on
     │                          │
     ▼                          ▼
┌─────────┐                  ┌──────┐
│  TRIPS  │ ◄────────────────│ ROUTES │
└─────────┘     follows      └──────┘
     │
     │ belongs to
```

```
       │
       ▼
┌─────────┐       submitted by    ┌──────────┐
│ REPORTS │ ◄─────────────────────│   USERS    │
└─────────┘                       │(Passengers)│
                                ┌──────────────┘
                                  │
                                  │ sends
                                  ▼
                                ┌──────────┐
                                │ MESSAGES │
                                └──────────┘
```

8. BACKUP & DISASTER RECOVERY

8.1 Backup Strategy

- **Frequency:** Daily automated backups at 2:00 AM IST
- **Retention:** 30 days rolling window
- **Storage:** AWS S3 with versioning
- **Testing:** Monthly restore tests

8.2 Point-in-Time Recovery

MongoDB Atlas supports point-in-time recovery up to 24 hours.

9. DATABASE PERFORMANCE OPTIMIZATION

9.1 Indexing Strategy

All collections have appropriate indexes (listed in each schema).

9.2 Query Optimization

- Use projection to retrieve only required fields
- Avoid full collection scans
- Use aggregation pipeline for complex queries

9.3 Connection Pooling

```
// MongoDB connection pool settings
minPoolSize: 10
maxPoolSize: 50
maxIdleTimeMS: 30000
```

10. TESTING STRATEGY

10.1 Unit Tests

- Password hashing/verification
- JWT token generation/validation
- Data validation rules

10.2 Integration Tests

- Database CRUD operations
- User authentication flow
- Report submission workflow

10.3 Security Tests

- SQL injection prevention
- XSS attack prevention
- CSRF token validation
- Rate limiting

11. ENVIRONMENT CONFIGURATION

```
# .env file
MONGODB_URI=mongodb+srv://user:pass@cluster.mongodb.net/where_is_my_bus
JWT_SECRET=your_super_secret_key_minimum_32_characters
JWT_ACCESS_EXPIRY=15m
JWT_REFRESH_EXPIRY=7d
BCRYPT_ROUNDS=12
AWS_S3_BUCKET=where-is-my-bus-photos
AWS_REGION=ap-south-1
```

12. MIGRATION PLAN

Phase 1: Database Setup (Week 1)

- Set up MongoDB Atlas cluster
- Create collections with schemas
- Create indexes
- Test connections

Phase 2: Authentication (Week 2)

- Implement JWT generation/verification
- Build login/register APIs
- Test authentication flow

Phase 3: CRUD Operations (Week 3-4)

- Build APIs for users, buses, routes
- Implement validation
- Test all CRUD operations

Phase 4: Integration (Week 5)

- Connect frontend to backend
- End-to-end testing
- Performance optimization

13. SUCCESS CRITERIA

✅ All collections created with proper schemas
✅ Indexes created for optimal query performance
✅ JWT authentication working
✅ Password hashing with BCrypt
✅ Role-based access control implemented
✅ Data validation working correctly
✅ Geospatial queries functional
✅ Backup strategy configured
✅ Security tests passing

Next Steps: Proceed to **Module 2: Core API Services** documentation.