element
zerolabs

Paradigm App – Backend Developer Technical Hands on Instruction

Company: Element Zero Labs

Author : Amit Yadav (Automation Engineer)

Must Have Before Starting

1. VS Code - Your main coding environment

```
# Install VS Code + these extensions:
- Node.js Extension Pack
- REST Client (for testing APIs)
- Prettier (code formatting)
```

2. Microphone + Camera - For team communication

- **Need clear audio/video for daily syncs and code reviews**
- **Test in browser before starting**

3. Stable Internet - For real-time development

- **Minimum: 25 Mbps down / 10 Mbps up**
- **Test: speedtest.net**

4. ngrok - Critical for n8n integration

```
# Install and setup
npm install -g ngrok
ngrok authtoken YOUR_TOKEN  # Get from ngrok.com
ngrok http 3000  # Creates: https://abc123.ngrok.io → localhost:3000
```

🏗️ Architecture Overview

```
Replit Frontend ←→ n8n Workflows ←→ Your VS Code Backend ←→ Database
(Team manages)   (Team manages)   (YOU build this)     (You setup)
```

Your Job: Build backend APIs that work with n8n

🔌 n8n Integration (What You Must Do)

1. Receive Webhooks FROM n8n

```
// n8n will POST to your backend
app.post('/api/webhooks/ai-response', (req, res) => {
  const { chat_id, ai_response } = req.body;
  // Save AI response to database
  // Send success response back
});
```

2. Expose Endpoints FOR n8n

```
// n8n will GET data from your backend
app.get('/api/tech/projects/:id/context', (req, res) => {
  // Return project data for AI processing
});
```

3. Make Backend Accessible via ngrok

```
# Step 1: Start your backend
npm run dev  # Running on localhost:3000

# Step 2: Start ngrok tunnel
ngrok http 3000  # Creates public URL

# Step 3: Give team your ngrok URL
# Example: https://abc123.ngrok.io
```

📋 Quick Setup Checklist

- **VS Code installed with extensions**
- **Microphone/camera working**
- **Internet speed tested (25+ Mbps)**
- **ngrok installed and tunnel working**
- **Backend server running on localhost:3000**
- **ngrok URL shared with team for n8n setup**

Part 1: Black API Services (User ID , role & permission , user Authentication & Developer Group Assignment & Permission)

Table of Contents - Part 1

---

1. Black Services Overview

1.1 Core Responsibilities

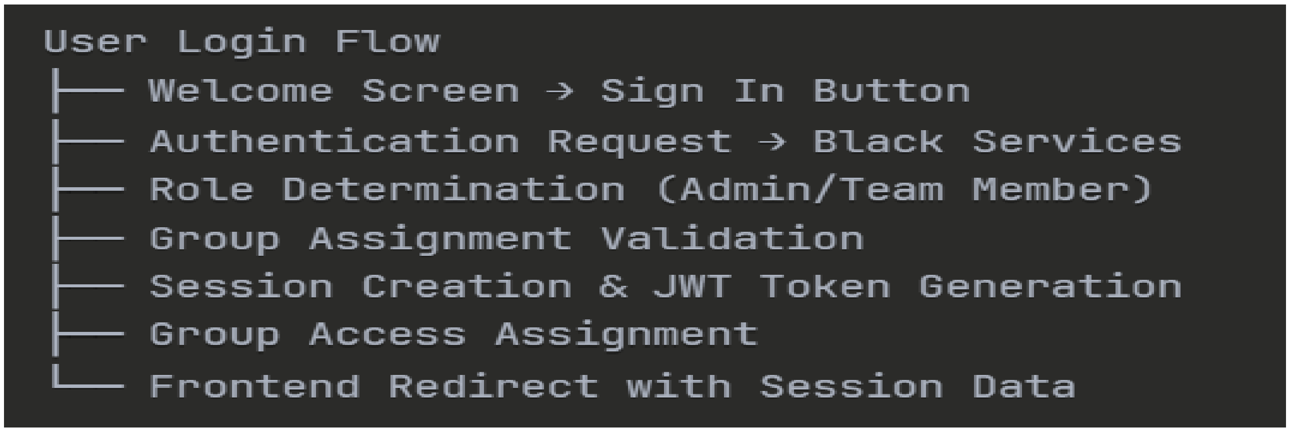
- **User Authentication: Email/password validation with secure hashing**
- **Role-Based Routing: Admin vs Team Member dashboard routing**
- **Group Assignment Management: Dynamic group access based on user roles**
- **Session Management: JWT token generation and validation**
- **Frontend Flow Control: Proper routing to admin dashboard or groups screen**

1.2 User Roles & Access

- **Admin/Super Admin: Access to "I AM AN ADMIN" screen → Groups management**
- **Team Member: Direct access to assigned Developer Groups**
- **Group Access: Admin/Super Admin (all 3 groups), Team Member (assigned group only)**

---

2. Authentication Flow Architecture



```
User Login Flow
├── Welcome Screen → Sign In Button
├── Authentication Request → Black Services
├── Role Determination (Admin/Team Member)
├── Group Assignment Validation
├── Session Creation & JWT Token Generation
├── Group Access Assignment
└── Frontend Redirect with Session Data
```

2.1 Complete Frontend Flow

2.2 Dashboard Routing Logic

```
// Authentication Response Routing
if (user.role === 'admin' || user.role === 'super_admin') {
  dashboard_route: "admin"  // → "I AM AN ADMIN" screen
} else if (user.role === 'team') {
  dashboard_route: "groups" // → Direct to Developer Groups

}
```

---

3. Authentication Database Schema

3.1 Users Table

```
// Collection: users

{

  "user_id": "PGM-346789/AY",

  "email": "amit.yadav@elementzerolabs.com",

  "password_hash": "$2b$10$hashed_password_here",

  "full_name": "Amit Yadav",

  "role": "admin", // "team", "admin", "super_admin"

  "permission": "Admin", // "Team", "Admin", "Super Admin"

  "department": "Automation Engineer",

  "status": "active",

  "assigned_groups": [

    "DGP/1-893267/MK",

    "DGP/2-891316/MK",

    "DGP/3-891134/MK"

  ],

  "created_at": "2025-01-01T00:00:00Z",

  "last_login": "2025-01-01T10:30:00Z",

  "login_count": 15,

  "failed_login_attempts": 0,

  "account_locked_until": null
}
```

3.2 User Sessions Table

```
{
  "session_id": "sess-341234/AY",
  "user_id": "PGM-346789/AY",
  "user_email": "amit.yadav@elementzerolabs.com",
  "user_role": "admin",
  "user_permission": "Admin",
  "assigned_groups": [
    "DGP/1-893267/MK",
    "DGP/2-891316/MK",
    "DGP/3-891134/MK"
  ],
  "session_start": "2025-01-01T10:30:00Z",
  "last_activity": "2025-01-01T10:45:00Z",
  "expires_at": "2025-01-02T10:30:00Z",
  "status": "active",
  "login_ip": "192.168.1.100",
  "user_agent": "Mozilla/5.0...",
  "jwt_token_hash": "sha256:token_hash_here"
}
```

3.3 Developer Groups Table

```
// Collection: developer_groups
{
  "group_id": "DGP/1-893267/MK",
  "group_name": "Paradigm/Dev-Group-1",
  "domain": "technical",
  "group_number": 1,
  "status": "active",
  "members": [
    {
      "user_id": "PGM-346789/AY",
      "role": "admin",
      "added_at": "2025-01-01T00:00:00Z"
    },
    {
      "user_id": "PGM-344722/MK",
      "role": "member",
      "added_at": "2025-01-01T00:00:00Z"
    }
  ],
  "created_at": "2025-01-01T00:00:00Z",
  "project_count": 0,
  "total_chats": 0
}
```

---

4. Black Services API Reference Table

API Endpoint	Method	Purpose	Access Level	Response Time Target
/api/auth/login	POST	User authentication & session creation	Public	< 200ms
/api/auth/logout	POST	Session termination & cleanup	Authenticated	< 100ms
/api/auth/validate-session	GET	Session validation& refresh	Authenticated	< 150ms
/api/auth/refresh-token	POST	JWT token refresh	Authenticated	< 100ms
/api/auth/group-access/{group_id}	GET	Group access validation	Authenticated	< 100ms
/api/auth/user-groups	GET	User's assigned groups list	Authenticated	< 150ms

5. Authentication API Implementation

5.1 User Login API

POST /api/auth/login

Purpose: Authenticate user and route to appropriate dashboard

```
// Request Body
{
  "email": "amit.yadav@elementzerolabs.com",
  "password": "SecurePass123"
}

// Response - Admin User
{
  "success": true,
  "user": {
    "user_id": "PGM-346789/AY",
    "email": "amit.yadav@elementzerolabs.com",
    "full_name": "Amit Yadav",
    "role": "admin",
    "permission": "Admin",
    "department": "Automation Engineer",
    "last_login": "2024-12-31T08:15:00Z",
    "login_count": 15
  },
  "session": {
    "session_id": "sess-341234/AY",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "expires_at": "2025-01-02T10:30:00Z",
    "refresh_token": "refresh_token_here"
  },
  "dashboard_route": "admin",
  "frontend_action": "show_admin_dashboard",
  "assigned_groups": [
    {
      "group_id": "DGP/1-893267/MK",
      "group_name": "Paradigm/Dev-Group-1",
      "access_level": "admin",
      "member_count": 3,
      "project_count": 2
    },
    {
      "group_id": "DGP/2-891316/MK",
      "group_name": "Paradigm/Dev-Group-2",
      "access_level": "admin",
      "member_count": 2,
      "project_count": 1
    },
    {
      "group_id": "DGP/3-891134/MK",
      "group_name": "Paradigm/Dev-Group-3",
      "access_level": "admin",
      "member_count": 1,
      "project_count": 0
    }
  ]
}

// Response - Team Member
{
  "success": true,
  "user": {
    "user_id": "PGM-349877/AB",
    "email": "amir@apostlefund.xyz",
    "full_name": "Amir Bershad",
    "role": "team",
    "permission": "Team",
    "department": "Social Media Lead",
    "last_login": "2024-12-30T14:20:00Z",
    "login_count": 8
  },
  "session": {
    "session_id": "sess-349999/AB",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "expires_at": "2025-01-02T10:30:00Z",
    "refresh_token": "team_refresh_token_here"
  },
  "dashboard_route": "groups",
  "frontend_action": "show_groups_directly",
  "assigned_groups": [
    {
      "group_id": "DGP/1-893267/MK",
      "group_name": "Paradigm/Dev-Group-1",
      "access_level": "member",
      "member_count": 3,
      "project_count": 2
    }
  ]
}
      {
      "group_id": "DGP/2-891316/MK",
      "group_name": "Paradigm/Dev-Group-2",
      "access_level": "admin",
      "member_count": 2,
      "project_count": 1
    },
    {
      "group_id": "DGP/3-891134/MK",
      "group_name": "Paradigm/Dev-Group-3",
      "access_level": "admin",
      "member_count": 1,
      "project_count": 0
    }
  ],
  "admin_dashboard": {
    "available_sections": [
      {
       "name": "Groups",
       "enabled": true,
       "action": "navigate_to_groups"
      },
      {
       "name": "I AM (Admin Console)",
       "enabled": false,
       "reason": "Feature not implemented"
      },
      {
       "name": "Identity & Organization",
       "enabled": false,
       "reason": "Feature not implemented"
      }
    ]
  }
}

// Response - Super Admin User
{
  "success": true,
  "user": {
    "user_id": "PGM-344722/MK",
    "email": "mikeal@electricoctopus.agency",
    "full_name": "Mikeal Kayanian",
    "role": "super_admin",
    "permission": "Super Admin",
    "department": "Founder/Super Admin",
    "last_login": "2024-12-31T09:30:00Z",
    "login_count": 25
},
"session": {
    "session_id": "sess-344999/MK",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.super_admin_token_payload.signature",
    "expires_at": "2025-01-02T10:30:00Z",
    "refresh_token": "super_admin_refresh_token"
},
"dashboard_route": "admin",
"frontend_action": "show_admin_dashboard",
"assigned_groups": [
    {
      "group_id": "DGP/1-893267/MK",
      "group_name": "Paradigm/Dev-Group-1",
      "access_level": "super_admin",
      "member_count": 3,
      "project_count": 2
    },
    {
      "group_id": "DGP/2-891316/MK",
      "group_name": "Paradigm/Dev-Group-2",
      "access_level": "super_admin",
      "member_count": 2,
      "project_count": 1
    },
    {
      "group_id": "DGP/3-891134/MK",
      "group_name": "Paradigm/Dev-Group-3",
      "access_level": "super_admin",
      "member_count": 1,
      "project_count": 0
    }
],
"admin_dashboard": {
    "available_sections": [
      {
        "name": "Groups",
        "enabled": true,
        "action": "navigate_to_groups"
      },
      {
        "name": "I AM (Admin Console)",
        "enabled": false,
        "reason": "Feature not implemented"
      },
      {
        "name": "Identity & Organization",
        "enabled": false,
        "reason": "Feature not implemented"
      }
    ]
  }
}

// Response - Team Member
{
  "success": true,
  "user": {
    "user_id": "PGM-349877/AB",
    "email": "amir@apostlefund.xyz",
    "full_name": "Amir Bershad",
    "role": "team",
    "permission": "Team",
    "department": "Social Media Lead",
    "last_login": "2024-12-30T14:20:00Z",
    "login_count": 8
  },
  "session": {
    "session_id": "sess-349999/AB",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "expires_at": "2025-01-02T10:30:00Z",
    "refresh_token": "team_refresh_token_here"
  },
  "dashboard_route": "groups",
  "frontend_action": "show_groups_directly",
  "assigned_groups": [
    {
      "group_id": "DGP/1-893267/MK",
      "group_name": "Paradigm/Dev-Group-1",
      "access_level": "member",
      "member_count": 3,
      "project_count": 2
    }
  ],
  "admin_dashboard": null
}

// Error Response - Invalid Credentials
{
  "success": false,
  "error": "Invalid email or password",
  "error_code": "INVALID_CREDENTIALS",
  "retry_allowed": true,
  "failed_attempts": 2,
  "max_attempts": 5,
  "frontend_action": "show_error_message"
```

```
}

// Error Response - Account Locked
{
  "success": false,
  "error": "Account temporarily locked due to multiple failed login attempts",
  "error_code": "ACCOUNT_LOCKED",
  "locked_until": "2025-01-01T11:00:00Z",
  "retry_allowed": false,
  "frontend_action": "show_lockout_message"
}
```

5.2 Session Validation API

GET /api/auth/validate-session

Purpose: Validate current session for frontend route protection

// Headers

```
// Headers

{

  "Authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."

}



// Response - Valid Admin Session

{

  "valid": true,

  "user": {

    "user_id": "PGM-346789/AY",

    "email": "amit.yadav@elementzerolabs.com",

    "full_name": "Amit Yadav",

    "role": "admin",
    "permission": "Admin",
    "assigned_groups": [
      "DGP/1-893267/MK",
      "DGP/2-891316/MK",
      "DGP/3-891134/MK"
    ]
  },
  "session": {
    "session_id": "sess-341234/AY",
    "expires_at": "2025-01-02T10:30:00Z",
    "time_remaining": "23h 45m",
    "last_activity": "2025-01-01T10:45:00Z"
  },
  "dashboard_route": "admin",
  "frontend_permissions": {
    "can_access_admin_dashboard": true,
    "can_access_all_groups": true,
    "can_manage_group_content": true
  }
}

// Response - Valid Team Member Session
{
  "valid": true,
  "user": {
    "user_id": "PGM-349877/AB",
    "email": "amir@apostlefund.xyz",
    "full_name": "Amir Bershad",
    "role": "team",
    "permission": "Team",
    "assigned_groups": [
      "DGP/1-893267/MK"
    ]
  },
  "session": {
    "session_id": "sess-349999/AB",
    "expires_at": "2025-01-02T10:30:00Z",
    "time_remaining": "23h 30m",
    "last_activity": "2025-01-01T10:30:00Z"
  },
  "dashboard_route": "groups",
  "frontend_permissions": {
    "can_access_admin_dashboard": false,
    "can_access_all_groups": false,
    "can_manage_group_content": true
  }
}

// Response - Invalid Session
{
  "valid": false,
  "error": "Session expired or invalid",
  "error_code": "INVALID_SESSION",
  "require_login": true,
  "frontend_action": "redirect_to_login"
}
```

5.3 Group Access Validation API

GET `/api/auth/group-access/{group_id}`

Purpose: Validate user access to specific developer group before entering

```
// Request
GET /api/auth/group-access/DGP/1-893267/MK

// Response - Admin Access
{
  "access_granted": true,
  "access_type": "admin",
  "user_info": {
    "user_id": "PGM-346789/AY",
    "role": "admin",
    "permission": "Admin"
```

```
},
"group_details": {
  "group_id": "DGP/1-893267/MK",
  "group_name": "Paradigm/Dev-Group-1",
  "domain": "technical",
  "group_number": 1,
  "member_count": 3,
  "project_count": 2,
  "total_chats": 15,
  "created_at": "2025-01-01T00:00:00Z"
},
"user_permissions": [
  "read",
  "write",
  "admin",
  "manage_users",
  "view_analytics",
  "manage_projects",
  "manage_knowledge_base",
  "create_projects",
  "delete_projects"
],
"frontend_access": {
  "can_enter_group": true,
  "can_create_projects": true,
    "can_manage_kb": true,
    "can_view_all_projects": true
  }
}

// Response - Team Member Access (Assigned Group)
{
  "access_granted": true,
  "access_type": "member",
  "user_info": {
    "user_id": "PGM-349877/AB",
    "role": "team",
    "permission": "Team"
  },
  "group_details": {
    "group_id": "DGP/1-893267/MK",
    "group_name": "Paradigm/Dev-Group-1",
    "domain": "technical",
    "member_count": 3,
    "project_count": 2
  },
  "user_permissions": [
    "read",
    "write",
    "create_projects",
    "upload_knowledge_base"
  ],
  "frontend_access": {
    "can_enter_group": true,
    "can_create_projects": true,
    "can_manage_kb": true,
    "can_view_own_projects": true
  }
}

// Response - Team Member Access Denied (Unassigned Group)
{
  "access_granted": false,
  "error": "User not assigned to this group",
  "error_code": "GROUP_ACCESS_DENIED",
  "user_info": {
    "user_id": "PGM-349877/AB",
    "role": "team",
    "assigned_groups_count": 1
  },
  "frontend_action": "show_access_denied_message",
  "suggestions": [
    "You only have access to Paradigm/Dev-Group-1",
    "Contact your administrator for additional group access"
  ]
```

```
}
```

---

6. Session Management APIs

6.1 User Logout API

POST /api/auth/logout

Purpose: Terminate session and redirect to welcome screen

```
// Headers
{
  "Authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "Content-Type": "application/json"
}
// Request Body (Optional)
{
  "logout_all_sessions": false
}
// Response
{
  "success": true,
  "message": "Logged out successfully",
  "session": {
    "session_id": "sess-341234/AY",
    "status": "terminated",
    "logout_time": "2025-01-01T11:00:00Z"
  },
  "frontend_action": "redirect_to_welcome",
  "sessions_terminated": 1
}
```

6.2 Token Refresh API

POST /api/auth/refresh-token

Purpose: Refresh JWT token without full re-authentication

```
// Request Body
{
  "refresh_token": "refresh_token_here"
}
// Response
{
  "success": true,
  "session": {
    "session_id": "sess-341234/AY",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.new_token_payload.signature",
    "expires_at": "2025-01-02T10:30:00Z",
    "refresh_token": "new_refresh_token_here"
  },
  "message": "Token refreshed successfully",
  "frontend_action": "update_session_token"
}
```

6.3 User Groups List API

GET /api/auth/user-groups

Purpose: Get list of groups for Developer Groups screen

```
// Headers
{
  "Authorization": "Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}

// Response - Admin User (All Groups)
{
  "success": true,
  "user_info": {
    "user_id": "PGM-346789/AY",
    "role": "admin",
    "permission": "Admin",
    "can_access_all_groups": true
  },
  "groups": [
    {
      "group_id": "DGP/1-893267/MK",
      "group_name": "Paradigm/Dev-Group-1",
      "display_name": "Paradigm/Dev-Group-1",
      "domain": "technical",
      "group_number": 1,
      "access_level": "admin",
      "member_count": 3,
      "project_count": 2,
      "last_activity": "2025-01-01T10:45:00Z",
  "can_enter": true,
  "frontend_style": "clickable-orange-button"
},
{
  "group_id": "DGP/2-891316/MK",
  "group_name": "Paradigm/Dev-Group-2",
  "display_name": "Paradigm/Dev-Group-2",
  "domain": "technical",
  "group_number": 2,
  "access_level": "admin",
  "member_count": 2,
  "project_count": 1,
  "last_activity": "2025-01-01T08:30:00Z",
  "can_enter": true,
  "frontend_style": "clickable-orange-button"
},
{
  "group_id": "DGP/3-891134/MK",
  "group_name": "Paradigm/Dev-Group-3",
  "display_name": "Paradigm/Dev-Group-3",
  "domain": "technical",
  "group_number": 3,
  "access_level": "admin",
  "member_count": 1,
  "project_count": 0,
      "last_activity": null,
      "can_enter": true,
      "frontend_style": "clickable-orange-button"
    }
  ],
  "total_groups": 3,
  "accessible_groups": 3
}

// Response - Team Member (Assigned Group Only)
{
  "success": true,
  "user_info": {
    "user_id": "PGM-349877/AB",
    "role": "team",
    "permission": "Team",
    "can_access_all_groups": false
  },
  "groups": [
    {
      "group_id": "DGP/1-893267/MK",
      "group_name": "Paradigm/Dev-Group-1",
      "display_name": "Paradigm/Dev-Group-1",
      "domain": "technical",
      "group_number": 1,
    "access_level": "member",
    "member_count": 3,
    "project_count": 2,
    "last_activity": "2025-01-01T10:45:00Z",
    "can_enter": true,
    "frontend_style": "clickable-orange-button"
   }
  ],
  "total_groups": 1,
  "accessible_groups": 1,
  "restricted_groups": [
   {
    "group_name": "Paradigm/Dev-Group-2",
    "reason": "Not assigned to this group"
   },
   {
    "group_name": "Paradigm/Dev-Group-3",
    "reason": "Not assigned to this group"
   }
  ]
}
```

---

7. Black Services Testing

7.1 Test Users Setup

// Test users for authentication testing

```
const testUsers = [
 {
   "user_id": "PGM-346789/AY",
   "email": "amit.yadav@elementzerolabs.com",
   "password": "SecurePass123", // Will be hashed
   "full_name": "Amit Yadav",
   "role": "admin",
   "permission": "Admin",
   "department": "Automation Engineer",
   "status": "active",
   "assigned_groups": [
     "DGP/1-893267/MK",
     "DGP/2-891316/MK",
     "DGP/3-891134/MK"
   ]
 },
{
   "user_id": "PGM-344722/MK",
   "email": "mikeal@electricoctopus.agency",
   "password": "AdminPass456", // Will be hashed
   "full_name": "Mikeal Kayanian",
   "role": "super_admin",
   "permission": "Super Admin",
   "department": "Founder/Super Admin",
   "status": "active",
    "assigned_groups": [
      "DGP/1-893267/MK",
      "DGP/2-891316/MK",
      "DGP/3-891134/MK"
    ]
  },
{
    "user_id": "PGM-349877/AB",
    "email": "amir@apostlefund.xyz",
    "password": "TeamPass789", // Will be hashed
    "full_name": "Amir Bershad",
    "role": "team",
    "permission": "Team",
    "department": "Social Media Lead",
    "status": "active",
    "assigned_groups": [
      "DGP/1-893267/MK"
    ]
  }
];
```

7.2 Frontend Flow Testing

```
const request = require('supertest');
const app = require('../../src/server');

describe('Black Services - Frontend Flow Testing', () => {

  describe('Authentication Flow Based on Role', () => {
    test('Admin login should route to admin dashboard', async () => {
      const loginData = {
        email: "amit.yadav@elementzerolabs.com",
        password: "SecurePass123"
      };

      const response = await request(app)
        .post('/api/auth/login')
        .send(loginData)
        .expect(200);

      // Validate admin dashboard routing
      expect(response.body.success).toBe(true);
      expect(response.body.user.role).toBe('admin');
      expect(response.body.dashboard_route).toBe('admin');
      expect(response.body.frontend_action).toBe('show_admin_dashboard');

      // Validate admin dashboard configuration
      expect(response.body.admin_dashboard).toBeDefined();
      expect(response.body.admin_dashboard.available_sections).toHaveLength(3);


        const groupsSection = response.body.admin_dashboard.available_sections.find(s => s.name ===
'Groups');
      expect(groupsSection.enabled).toBe(true);
      expect(groupsSection.action).toBe('navigate_to_groups');


        const adminConsole = response.body.admin_dashboard.available_sections.find(s => s.name === 'I
AM (Admin Console)');
      expect(adminConsole.enabled).toBe(false);
      expect(adminConsole.reason).toBe('Feature not implemented');
    });


    test('Super Admin login should route to admin dashboard (same as admin)', async () => {
      const loginData = {
        email: "mikeal@electricoctopus.agency",
        password: "AdminPass456"
      };


      const response = await request(app)
        .post('/api/auth/login')
        .send(loginData)
        .expect(200);


      expect(response.body.success).toBe(true);
      expect(response.body.user.role).toBe('super_admin');
      expect(response.body.dashboard_route).toBe('admin');
      expect(response.body.frontend_action).toBe('show_admin_dashboard');
      expect(response.body.admin_dashboard).toBeDefined();
    });

    test('Team member login should route directly to groups', async () => {
      const loginData = {
        email: "amir@apostlefund.xyz",
        password: "TeamPass789"
      };

      const response = await request(app)
        .post('/api/auth/login')
        .send(loginData)
        .expect(200);

      expect(response.body.success).toBe(true);
      expect(response.body.user.role).toBe('team');
      expect(response.body.dashboard_route).toBe('groups');
      expect(response.body.frontend_action).toBe('show_groups_directly');
      expect(response.body.admin_dashboard).toBeNull();

      // Team member should only have access to one group
      expect(response.body.assigned_groups).toHaveLength(1);
```

```
    expect(response.body.assigned_groups[0].group_name).toBe('Paradigm/Dev-Group-1');
  });
});

describe('Developer Groups Screen Data', () => {
  let adminToken, teamToken;

  beforeAll(async () => {
    // Get admin token
    const adminLogin = await request(app)
      .post('/api/auth/login')
      .send({
        email: "amit.yadav@elementzerolabs.com",
        password: "SecurePass123"
      });
    adminToken = adminLogin.body.session.token;

    // Get team member token
    const teamLogin = await request(app)
      .post('/api/auth/login')
      .send({
        email: "amir@apostlefund.xyz",
        password: "TeamPass789"
      });
    teamToken = teamLogin.body.session.token;
```

```
});

test('Admin should see all 3 developer groups as clickable', async () => {
  const response = await request(app)
    .get('/api/auth/user-groups')
    .set('Authorization', `Bearer ${adminToken}`)
    .expect(200);

  expect(response.body.success).toBe(true);
  expect(response.body.groups).toHaveLength(3);
  expect(response.body.accessible_groups).toBe(3);

  // Validate all groups are accessible
  response.body.groups.forEach(group => {
    expect(group.can_enter).toBe(true);
    expect(group.access_level).toBe('admin');
    expect(group.frontend_style).toBe('clickable-orange-button');
    expect(group.group_name).toMatch(/Paradigm\/Dev-Group-[1-3]/);
  });
});

test('Team member should see only assigned group as clickable', async () => {
  const response = await request(app)
    .get('/api/auth/user-groups')
    .set('Authorization', `Bearer ${teamToken}`)
```

```
      .expect(200);

    expect(response.body.success).toBe(true);
    expect(response.body.groups).toHaveLength(1);
    expect(response.body.accessible_groups).toBe(1);
    expect(response.body.restricted_groups).toHaveLength(2);

    const accessibleGroup = response.body.groups[0];
    expect(accessibleGroup.group_name).toBe('Paradigm/Dev-Group-1');
    expect(accessibleGroup.can_enter).toBe(true);
    expect(accessibleGroup.access_level).toBe('member');
    expect(accessibleGroup.frontend_style).toBe('clickable-orange-button');
  });
});

describe('Group Access Validation for Frontend', () => {
  let adminToken, teamToken;

  beforeAll(async () => {
    const adminLogin = await request(app)
      .post('/api/auth/login')
      .send({
        email: "amit.yadav@elementzerolabs.com",
        password: "SecurePass123"
      });
  adminToken = adminLogin.body.session.token;

  const teamLogin = await request(app)
    .post('/api/auth/login')
    .send({
      email: "amir@apostlefund.xyz",
      password: "TeamPass789"
    });
  teamToken = teamLogin.body.session.token;
});

test('Admin can access all groups from frontend', async () => {
  const groupIds = [
    'DGP/1-893267/MK',
    'DGP/2-891316/MK',
    'DGP/3-891134/MK'
  ];

  for (const groupId of groupIds) {
    const response = await request(app)
      .get(`/api/auth/group-access/${encodeURIComponent(groupId)}`)
      .set('Authorization', `Bearer ${adminToken}`)
      .expect(200);

    expect(response.body.access_granted).toBe(true);
    expect(response.body.access_type).toBe('admin');
    expect(response.body.frontend_access.can_enter_group).toBe(true);
    expect(response.body.frontend_access.can_create_projects).toBe(true);
    expect(response.body.frontend_access.can_manage_kb).toBe(true);
  }
});

test('Team member can only access assigned group', async () => {
  const assignedGroupId = 'DGP/1-893267/MK';

  const response = await request(app)
    .get(`/api/auth/group-access/${encodeURIComponent(assignedGroupId)}`)
    .set('Authorization', `Bearer ${teamToken}`)
    .expect(200);

  expect(response.body.access_granted).toBe(true);
  expect(response.body.access_type).toBe('member');
  expect(response.body.frontend_access.can_enter_group).toBe(true);
  expect(response.body.frontend_access.can_create_projects).toBe(true);
});

test('Team member should be denied access to unassigned groups', async () => {
  const unassignedGroupIds = [
    'DGP/2-891316/MK',
    'DGP/3-891134/MK'
    ];
    for (const groupId of unassignedGroupIds) {
      const response = await request(app)
        .get(`/api/auth/group-access/${encodeURIComponent(groupId)}`)
        .set('Authorization', `Bearer ${teamToken}`)
        .expect(403);
      expect(response.body.access_granted).toBe(false);
      expect(response.body.error_code).toBe('GROUP_ACCESS_DENIED');
      expect(response.body.frontend_action).toBe('show_access_denied_message');
      expect(response.body.suggestions).toBeDefined();
    }
  });
});

describe('Session Management for Frontend', () => {
  test('Session validation should return proper frontend routing info', async () => {
    const loginResponse = await request(app)
      .post('/api/auth/login')
      .send({
        email: "amit.yadav@elementzerolabs.com",
        password: "SecurePass123"
      });
    const validationResponse = await request(app)
      .get('/api/auth/validate-session')
      .set('Authorization', `Bearer ${loginResponse.body.session.token}`)
```

```
        .expect(200);
      expect(validationResponse.body.valid).toBe(true);
      expect(validationResponse.body.dashboard_route).toBe('admin');
      expect(validationResponse.body.frontend_permissions).toBeDefined();
      expect(validationResponse.body.frontend_permissions.can_access_admin_dashboard).toBe(true);
    });
    test('Logout should provide proper frontend redirect action', async () => {
      const loginResponse = await request(app)
        .post('/api/auth/login')
        .send({
          email: "amit.yadav@elementzerolabs.com",
          password: "SecurePass123"
        });
      const logoutResponse = await request(app)
        .post('/api/auth/logout')
        .set('Authorization', `Bearer ${loginResponse.body.session.token}`)
        .expect(200);
      expect(logoutResponse.body.success).toBe(true);
      expect(logoutResponse.body.frontend_action).toBe('redirect_to_welcome');
    });
  });
});
```

7.3 CORS and Replit Integration Testing

```
// tests/auth/replit-integration.test.js
```

```
const request = require('supertest');
const app = require('../../src/server');

describe('Black Services - Replit Integration', () => {

  test('Should handle CORS for Replit frontend', async () => {
    const replit_url = process.env.FRONTEND_URL || 'https://paradigm-frontend.replit.dev';

    const response = await request(app)
      .options('/api/auth/login')
      .set('Origin', replit_url)
      .expect(200);

    expect(response.headers['access-control-allow-origin']).toBe(replit_url);
    expect(response.headers['access-control-allow-methods']).toContain('POST');
    expect(response.headers['access-control-allow-headers']).toContain('Authorization');
  });

  test('Should accept authentication requests from Replit', async () => {
    const replit_url = process.env.FRONTEND_URL || 'https://paradigm-frontend.replit.dev';

    const response = await request(app)
      .post('/api/auth/login')
      .set('Origin', replit_url)
      .send({
```

```
      email: "amit.yadav@elementzerolabs.com",

      password: "SecurePass123"

    })

    .expect(200);


  expect(response.body.success).toBe(true);

  expect(response.headers['access-control-allow-origin']).toBe(replit_url);

});


test('Should validate sessions from Replit with proper headers', async () => {

  const replit_url = process.env.FRONTEND_URL || 'https://paradigm-frontend.replit.dev';


  // First login

  const loginResponse = await request(app)

    .post('/api/auth/login')

    .set('Origin', replit_url)

    .send({

      email: "amit.yadav@elementzerolabs.com",

      password: "SecurePass123"

    });

  // Then validate session

  const validationResponse = await request(app)

    .get('/api/auth/validate-session')

    .set('Authorization', `Bearer ${loginResponse.body.session.token}`)

    .set('Origin', replit_url)
```

```
      .expect(200);

    expect(validationResponse.body.valid).toBe(true);

    expect(validationResponse.headers['access-control-allow-origin']).toBe(replit_url);

  });

});
```

---

8. VS Code Development Setup

8.1 Project Structure for Black Services

```
paradigm-backend/
├── src/
│   ├── services/
│   │   └── black-services/
│   │       ├── auth-controller.js       // Main authentication logic
│   │       ├── session-manager.js       // Session handling
│   │       ├── group-access-controller.js // Group access validation
│   │       └── frontend-router.js       // Frontend routing logic
│   ├── middleware/
│   │   ├── auth-middleware.js         // JWT validation
│   │   ├── cors-handler.js           // CORS for Replit
│   │   ├── rate-limiter.js           // Login rate limiting
│   │   └── validation-middleware.js    // Input validation
│   ├── database/
│   │   ├── oracle-nosql/
│   │   │   ├── connection.js          // Database connection
│   │   │   ├── user-operations.js      // User CRUD operations
│   │   │   ├── session-operations.js    // Session management
│   │   │   └── group-operations.js     // Group data operations
│   └── utils/
│       ├── uuid-generator.js          // Dynamic UUID generation
│       ├── password-hasher.js          // Bcrypt password handling
│       ├── jwt-manager.js             // JWT token management
│       └── frontend-response-formatter.js // Response formatting
│
```

```
├── tests/
│   └── auth/
│       ├── frontend-flow.test.js       // Frontend flow testing
│       ├── authentication.test.js      // Auth API testing
│       ├── group-access.test.js        // Group access testing
│       ├── session-management.test.js  // Session API testing
│       ├── replit-integration.test.js  // CORS and integration
│       └── performance.test.js         // Performance testing
├── config/
│   ├── environment.js              // Environment configuration
│   ├── database.js                 // Database configuration
│   ├── cors.js                     // CORS configuration
│   └── security.js                 // Security settings
└── scripts/
    ├── setup-auth-database.js         // Setup auth tables
    ├── seed-test-users.js             // Seed test data
    ├── reset-auth-data.js             // Reset for testing

    └── validate-replit-connection.js  // Test Replit connection
```

8.2 Environment Configuration

```
// config/environment.js
module.exports = {
 // Server Configuration
 PORT: process.env.PORT || 3000,
 NODE_ENV: process.env.NODE_ENV || 'development',

 // Database Configuration
 ORACLE_NOSQL: {
  endpoint: process.env.ORACLE_NOSQL_ENDPOINT || 'http://localhost:8080',
  compartment: process.env.ORACLE_NOSQL_COMPARTMENT || 'dev-compartment',
  region: 'us-ashburn-1',
  timeout: 30000
 },

 // Frontend Configuration (Replit)
 FRONTEND: {
  url: process.env.FRONTEND_URL || 'https://paradigm-frontend.replit.dev',
  allowed_origins: [
   process.env.FRONTEND_URL || 'https://paradigm-frontend.replit.dev',
   'http://localhost:3001' // For local development
  ]
 },

 // Authentication Configuration
 JWT: {
    secret: process.env.JWT_SECRET || 'paradigm-dev-jwt-secret-2024',
    expiry: '24h',
    refresh_expiry: '7d'
  },

  // Security Configuration
  SECURITY: {
    max_login_attempts: 5,
    lockout_duration: 3600000, // 1 hour in milliseconds
    bcrypt_rounds: 10,
    rate_limit_requests: 100,
    rate_limit_window: 900000 // 15 minutes
  },

  // Logging Configuration
  LOGGING: {
    level: 'debug',
    file: './logs/paradigm-auth.log',
    console: true
  }
};
```

8.3 Development Commands

```
// package.json scripts
{
  "scripts": {
    "start": "node src/server.js",
    "dev": "nodemon src/server.js",
    "dev:auth": "nodemon src/server.js --inspect",

    "setup:auth": "node scripts/setup-auth-database.js",
    "seed:test-users": "node scripts/seed-test-users.js",
    "reset:auth": "node scripts/reset-auth-data.js",
    "validate:replit": "node scripts/validate-replit-connection.js",

    "test": "jest --watchAll",
    "test:auth": "jest tests/auth --verbose",
    "test:auth:flow": "jest tests/auth/frontend-flow.test.js --verbose",
    "test:auth:login": "jest tests/auth/authentication.test.js --verbose",
    "test:auth:groups": "jest tests/auth/group-access.test.js --verbose",
    "test:auth:session": "jest tests/auth/session-management.test.js --verbose",
    "test:auth:replit": "jest tests/auth/replit-integration.test.js --verbose",
    "test:auth:performance": "jest tests/auth/performance.test.js --verbose",

    "lint": "eslint src/",
    "lint:fix": "eslint src/ --fix"
  }
}
```

8.4 Testing Workflow Commands

```bash
# Initial Setup
npm install
npm run setup:auth
npm run seed:test-users

# Development Testing
npm run dev:auth             # Start with auth debugging
npm run validate:replit       # Test Replit connection

# Authentication Testing
npm run test:auth:flow         # Test complete frontend flow
npm run test:auth:login        # Test login functionality
npm run test:auth:groups        # Test group access
npm run test:auth:session       # Test session management
npm run test:auth:replit        # Test Replit integration

# Performance Testing
npm run test:auth:performance     # Test response times

# Reset for Fresh Testing
npm run reset:auth             # Reset all auth data

npm run seed:test-users         # Re-seed test users
```

8.5 Testing Checklist for Black Services

Frontend Flow Testing:

- Admin login routes to "I AM AN ADMIN" screen
- Super Admin login routes to "I AM AN ADMIN" screen
- Team member login routes directly to Developer Groups
- "Groups" button in admin screen is enabled
- Other admin screen buttons are disabled
- Developer Groups screen shows correct groups per user role

Authentication API Testing:

- Valid credentials return proper dashboard routing
- Invalid credentials are rejected with proper error codes
- Session tokens are generated correctly
- User roles are validated and assigned properly
- Group assignments are returned correctly

Group Access Testing:

- Admin/Super Admin can access all 3 groups
- Team member can only access assigned group (Group-1)
- Group access validation returns proper frontend permissions
- Unassigned group access is properly denied

Session Management Testing:

- Session validation works for route protection
- Token refresh functionality works
- Logout properly terminates sessions
- Session expiration is handled correctly

Replit Integration Testing:

- CORS is configured correctly for Replit frontend
- Authentication requests from Replit are accepted
- Session validation works with Replit origin
- Response headers include proper CORS information

Performance Testing:

- Login API responds within 200ms
- Session validation responds within 150ms
- Group access validation responds within 100ms
- Concurrent login requests are handled properly (10+ users)

Security Testing:

- Password hashing is implemented correctly
- JWT tokens are secure and properly formatted
- Rate limiting prevents brute force attacks
- Input validation prevents injection attacks
- Account lockout works after failed attempts

Key Features:

✅ System-generated System ID generation formats (PGM-346789/AY, DGP/1-893267/MK)
✅ Role-based dashboard routing (Admin → "I AM AN ADMIN", Team → Groups)
✅ Group access validation for Developer Groups 1-3
✅ JWT session management with refresh tokens
✅ CORS configuration for Replit integration
✅ Comprehensive testing suite for all scenarios
✅ Oracle NoSQL database integration
✅ Complete VS Code development setup

Paradigm App - Green Services Implementation Guide

Developer Group Backend APIs

GROUP CHAT MANAGEMENT:

1. /api/tech/chat/group/create | POST | NO MCP | < 200ms | Create group chat (direct n8n)
2. /api/tech/chat/group/send | POST | NO MCP | < 2000ms | Send message (bypass MCP)
3. /api/tech/chat/group/history/{chat_id} | GET | NO MCP | < 200ms | Get group chat history
4. /api/tech/chat/group/list/{group_id} | GET | NO MCP | < 250ms | List user's group chats

PROJECT MANAGEMENT:

5. /api/tech/projects/create | POST | NO MCP | < 300ms | Create new project with system UUID

6. /api/tech/projects/list/{group_id} | GET | NO MCP | < 200ms | List projects (role-based filtering)

7. /api/tech/projects/details/{project_id} | GET | NO MCP | < 150ms | Get project details with KB list

8. /api/tech/projects/delete/{project_id} | DELETE | CONDITIONAL | < 400ms | Delete project and cleanup

KNOWLEDGE BASE MANAGEMENT:

9./api/tech/kb/upload | POST | MCP CRITICAL | < 500ms | Upload KB file (additive)

10. /api/tech/kb/status/{kb_id} | GET | NO MCP | < 100ms | Check KB processing status

11. /api/tech/kb/list/{project_id} | GET | NO MCP | < 150ms | List all KB files for project

12. /api/tech/kb/delete/{kb_id} | DELETE | MCP CRITICAL | < 300ms | Delete KB file

INSTRUCTION MANAGEMENT:

13. /api/tech/instructions/create | POST | MCP CRITICAL | < 300ms | Create instruction for KB (one per KB)

14. /api/tech/instructions/update/{instruction_id} | PUT | MCP CRITICAL | < 250ms | Update existing instruction

15. /api/tech/instructions/get/{kb_id} | GET | NO MCP | < 150ms | Get instruction for specific KB

16. /api/tech/instructions/delete/{instruction_id} | DELETE | MCP CRITICAL | < 200ms | Delete instruction

PROJECT CHAT MANAGEMENT:

17. /api/tech/chat/project/create | POST | MCP CRITICAL | < 400ms | Create project chat with KB selection

18. /api/tech/chat/project/send | POST | MCP CRITICAL | < 2500ms | Send message (via Technical MCP)

19. /api/tech/chat/project/history/{chat_id} | GET | NO MCP | < 300ms | Get project chat history

20. /api/tech/chat/project/list/{project_id} | GET | NO MCP | < 200ms | List all chats for project

21. /api/tech/chat/kb-selection/update | POST | MCP CRITICAL | < 400ms | Update KB selection (before first message)

FILE UPLOAD MANAGEMENT:

22. /api/tech/files/upload | POST | CONDITIONAL | < 1000ms | Upload files to chats

23. /api/tech/files/list/{chat_id} | GET | NO MCP | < 150ms | List files in chat

24. /api/tech/files/delete/{file_id} | DELETE | NO MCP | < 200ms | Delete uploaded file

GROUP CHAT HISTORY MANAGEMENT:

25. /api/tech/history/group-chat/create | POST | NO MCP | < 200ms | Create group chat history record

26. /api/tech/history/group-chat/list/{chat_id} | GET | NO MCP | < 150ms | List group chat history

PROJECT CHAT HISTORY MANAGEMENT:

27. /api/tech/history/project-chat/create | POST | NO MCP | < 200ms | Create project chat history record

28. /api/tech/history/project-chat/list/{chat_id} | GET | NO MCP | < 150ms | List project chat history

SYSTEM MONITORING:

MCP Classifications:

- **MCP CRITICAL (9 APIs): Must route through Technical MCP**
- **NO MCP (18 APIs): Direct operations, no AI processing**
- **CONDITIONAL (3 APIs): MCP involvement depends on context**

Database Schema (5 Core Tables)

1. Projects Table

```
{
 "project_id": "PRJ-936921/DGP-1/AY",
 "project_name": "Node.js API Framework",
 "project_description": "Building secure REST API with authentication",
 "group_id": "DGP/1-893267/MK",
 "created_by": "PGM-346789/AY",
 "knowledge_bases": ["kwb-360199/PRJ-936921/DGP-1/AY"],
 "chat_count": 0,
 "created_at": "2025-01-01T10:30:00Z",
 "updated_at": "2025-01-01T10:30:00Z"
}
```

2. Knowledge Bases Table

```
{
 "kb_id": "kwb-360199/PRJ-936921/DGP-1/AY",
 "kb_name": "Express.js Security Guide",
 "kb_description": "Best practices for Node.js security",
 "project_id": "PRJ-936921/DGP-1/AY",
 "group_id": "DGP/1-893267/MK",
 "created_by": "PGM-346789/AY",
 "file_type": "pdf",
 "upload_status": "ready", // "processing", "ready", "failed"
 "mcp_processed": true,
 "additive_sequence": 1,
 "created_at": "2025-01-01T10:30:00Z"
}
```

3. Instructions Table

```
{
 "instruction_id": "INN-278091/kwb-360199/PRJ-936921/DGP-1/AB",
 "kb_id": "kwb-360199/PRJ-936921/DGP-1/AY",
 "project_id": "PRJ-936921/DGP-1/AY",
 "group_id": "DGP/1-893267/MK",
 "created_by": "PGM-349877/AB",
 "instruction_name": "API Security Implementation",
 "instruction_content": "1. Validate all inputs 2. Use HTTPS everywhere...",
 "created_at": "2025-01-01T10:30:00Z",
 "updated_at": "2025-01-01T10:30:00Z"
}
```

4. Project Chat Messages Table

```
{
 "message_id": "msg-518091/PRJ-936921/DGP-1/AY",
 "chat_id": "cha-518091/PRJ-936921/DGP-1/AY",
 "chat_type": "project",
 "project_id": "PRJ-936921/DGP-1/AY",
 "group_id": "DGP/1-893267/MK",
 "user_id": "PGM-346789/AY",
 "selected_kb_id": "kwb-360199/PRJ-936921/DGP-1/AY",
 "user_message": "How do I implement middleware security?",
 "ai_response": "Based on the security guide...",
 "response_metadata": {
  "mcp_processed": true,
  "kb_references": ["Section 4.3", "Page 15"],
  "processing_time": 2400
 },
 "created_at": "2025-01-01T10:30:00Z"
}
```

5. Group Chat Messages Table

```
{
  "message_id": "msg-766081/DGP-1/AY",
  "chat_id": "cha-766081/DGP-1/AY",
  "chat_type": "group",
  "group_id": "DGP/1-893267/MK",
  "user_id": "PGM-346789/AY",
  "user_message": "What are ES2024 features?",
  "ai_response": "ES2024 introduces several new features...",
  "response_metadata": {
    "mcp_processed": false,
    "direct_n8n": true,
    "processing_time": 1800
  },
  "created_at": "2025-01-01T10:30:00Z"
}
```

6. Chat Files Table

```
{
  "file_id": "fil-pdf-766081/DGP-1/AY", // Group chat file
  "chat_id": "cha-766081/DGP-1/AY",
  "chat_type": "group", // or "project"
  "group_id": "DGP/1-893267/MK",
  "project_id": null, // null for group chats
  "user_id": "PGM-346789/AY",
  "file_name": "coding-standards.pdf",
  "file_type": "pdf",
  "upload_status": "ready",
  "created_at": "2025-01-01T10:30:00Z"
}
```

7. Group Chat History Table

```
{
  "history_id": "his-XX1234/cha-766081/DGP-1/AY",
  "parent_chat_id": "cha-766081/DGP-1/AY",
  "chat_type": "group",
  "group_id": "DGP/1-893267/MK",
  "user_id": "PGM-346789/AY",
  "message_content": "Chat message content",
  "message_timestamp": "2025-01-01T10:30:00Z",
  "created_at": "2025-01-01T10:30:00Z"
}
```

8. Project Chat History Table

```
{
  "history_id": "his-XX9012/cha-518091/PRJ-936921/DGP-1/AY",
  "parent_chat_id": "cha-518091/PRJ-936921/DGP-1/AY",
  "chat_type": "project",
  "project_id": "PRJ-936921/DGP-1/AY",
  "group_id": "DGP/1-893267/MK",
  "user_id": "PGM-346789/AY",
  "selected_kb_id": "kwb-360199/PRJ-936921/DGP-1/AY",
  "message_content": "Enhanced message with KB context",
  "kb_references": ["Section 4.3"],
  "message_timestamp": "2025-01-01T10:30:00Z",
  "created_at": "2025-01-01T10:30:00Z"
}
```

Critical Implementation Rules

⚠️ KB Selection Rules (Warning-1 & Warning-2)

```
// Rule 1: Only ONE KB per chat (or none)
if (selected_kbs.length > 1) {
  return error("Only one KB can be selected per chat");
}

// Rule 2: KB selection LOCKED after first message
if (chat.message_count > 0 && kb_change_requested) {
  return error("Cannot change KB selection after chat has begun");
}
```

📋 Instruction Rules (Critical)

```
// Rule 3: Only ONE instruction per KB maximum
if (kb_has_instruction && new_instruction_requested) {
  return error("Knowledge base already has an instruction. Update existing one.");
}

// Rule 4: Instructions are editable
if (instruction_exists && update_requested) {
  // Allow updating instruction content, name, and description
  return updateInstruction(instruction_id, new_content);
}
```

🗂️ File UID Generation (Important-1 & Important-2)

```
// Group chat files - inherit chat serial number
const groupFileId = `fil-${fileType}-${chatSerialNumber}/${groupPath}/${userInitials}`;
// Example: fil-pdf-766081/DGP-1/AY
// Route: Green Service → n8n → Anthropic

// Project chat files - inherit chat serial number
const                                                   projectFileId                                          =
`fil-${fileType}-${chatSerialNumber}/PRJ-${projectSerial}/${groupPath}/${userInitials}`;
// Example: fil-docx-518091/PRJ-936921/DGP-1/AY
// Route: Green Service → Technical MCP → n8n → Anthropic
```

🔒 Chat Isolation (Important)

```
// Group chats CANNOT move to projects
if (chat_type === 'group' && move_to_project_requested) {
  return error("Group chats cannot be moved to projects");
}
```

System UUID Generation Patterns

User & Group IDs (Black Services)

```
// Users
"PGM-346789/AY", "PGM-344722/MK", "PGM-349877/AB"

// Groups
"DGP/1-893267/MK", "DGP/2-891316/MK", "DGP/3-891134/MK"
```

Activity IDs (Green Services)

```
// Projects
"PRJ-936921/DGP-1/AY", "PRJ-932313/DGP-1/MK", "PRJ-939717/DGP-1/MK"

// Group Chats
"cha-766081/DGP-1/AY", "cha-762933/DGP-1/MK", "cha-769157/DGP-1/AB"

// Project Chats
"cha-518091/PRJ-936921/DGP-1/AY", "cha-517321/PRJ-936921/DGP-1/MK"
```

```
// Knowledge Bases
"kwb-360199/PRJ-936921/DGP-1/AY", "kwb-361311/PRJ-939771/DGP-3/AB"

// Instructions
"INN-278091/kwb-360199/PRJ-936921/DGP-1/AB"

// Files (inherit chat serial numbers)
"fil-pdf-766081/DGP-1/AY"          // Group chat file
"fil-docx-518091/PRJ-936921/DGP-1/AY"  // Project chat file

// Chat History
"his-XX1234/cha-766081/DGP-1/AY"       // Group chat history
"his-XX9012/cha-518091/PRJ-936921/DGP-1/AY"  // Project chat history
```

n8n Integration Routes

Technical MCP Routes (MCP CRITICAL APIs)

```
const mcpRoutes = {
 projectChat: '/webhook/tech-mcp/project-chat',
 kbUpload: '/webhook/tech-mcp/kb-upload',
 kbDelete: '/webhook/tech-mcp/kb-delete',
 kbContextBuild: '/webhook/tech-mcp/context-build',
 instructionCreate: '/webhook/tech-mcp/instruction-create',
 instructionUpdate: '/webhook/tech-mcp/instruction-update',
 instructionDelete: '/webhook/tech-mcp/instruction-delete'
};
```

Direct Anthropic Routes (NO MCP APIs)

```
const directRoutes = {
 groupChat: '/webhook/anthropic/group-chat',
 groupFile: '/webhook/anthropic/group-file'
};
```

Development Environment Setup

```
# Install dependencies
npm install express axios uuid multer

# Environment variables
N8N_BASE_URL=http://localhost:5678
ORACLE_NOSQL_ENDPOINT=http://localhost:8080
JWT_SECRET=paradigm-dev-secret

# Start development
npm run dev

# Test commands
npm run test:green-services      # Test all 30 APIs
npm run test:group-chat         # Test direct n8n routing
npm run test:project-chat       # Test MCP integration
npm run test:kb-management      # Test KB upload/processing
npm run test:instruction-management # Test instruction CRUD
npm run test:chat-history       # Test history functionality
```

Key API Examples

Create Group Chat (NO MCP)

```
POST /api/tech/chat/group/create
{
  "group_id": "DGP/1-893267/MK",
  "initial_message": "What are Node.js best practices?"
}
// Response: Direct n8n → Anthropic (fast, no KB context)
// Chat ID: cha-766081/DGP-1/AY
```

Create Project Chat (MCP CRITICAL)

```
POST /api/tech/chat/project/create
{
  "project_id": "PRJ-936921/DGP-1/AY",
  "selected_kb_id": "kwb-360199/PRJ-936921/DGP-1/AY", // Optional, only ONE allowed
  "initial_message": "How do I implement security middleware?"
}
// Response: Technical MCP → n8n → Anthropic (enhanced with KB context)
// Chat ID: cha-518091/PRJ-936921/DGP-1/AY
```

Upload KB (MCP CRITICAL)

```
POST /api/tech/kb/upload (multipart/form-data)
{
  "project_id": "PRJ-936921/DGP-1/AY",
  "kb_name": "Security Guide",
  "kb_description": "Best practices for API security",
  "file": [binary data]
}
// Response: KB ID: kwb-360199/PRJ-936921/DGP-1/AY
// Triggers Technical MCP processing, additive to existing KBs
```

Create Instruction (MCP CRITICAL)

```
POST /api/tech/instructions/create
{
  "kb_id": "kwb-360199/PRJ-936921/DGP-1/AY",
  "instruction_name": "API Security Implementation",
   "instruction_content": "1. Validate all inputs\n2. Use HTTPS everywhere\n3. Implement
rate limiting..."
}
// Response: Instruction ID: INN-278091/kwb-360199/PRJ-936921/DGP-1/AB
// Note: Only ONE instruction allowed per KB
```

Upload File to Chat

```
POST /api/tech/files/upload (multipart/form-data)
{
  "chat_id": "cha-766081/DGP-1/AY", // Group chat
  "file": [binary data]
}
// Response: File ID: fil-pdf-766081/DGP-1/AY (inherits chat serial number)

POST /api/tech/files/upload (multipart/form-data)
{
  "chat_id": "cha-518091/PRJ-936921/DGP-1/AY", // Project chat
  "file": [binary data]
}
// Response: File ID: fil-docx-518091/PRJ-936921/DGP-1/AY (inherits chat serial number)
```

Testing Checklist

Core Functionality:

- **All 30 APIs respond within target times**
- **System UUID generation follows collision-free patterns**
- **KB selection rules enforced (one per chat, locked after first message)**
- **Instruction rules enforced (one per KB, editable)**
- **File upload UIDs generated correctly with chat serial inheritance**
- **MCP routing works for critical APIs**
- **Direct n8n routing works for group chats**
- **Chat isolation maintained (group ↔ project)**

UUID System Testing:

- **User IDs: PGM-{prefix}{4-digit}/{initials} format**
- **Group IDs: DGP/{group-num}-{prefix}{4-digit}/{creator} format**
- **Project IDs: PRJ-{prefix}{4-digit}/DGP-{group#}/{user} format**
- **Group Chat IDs: cha-{prefix}{4-digit}/DGP-{group#}/{user} format**
- **Project Chat IDs: cha-{prefix}{4-digit}/PRJ-{project-serial}/DGP-{group#}/{user} format**
- **KB IDs: kwb-{prefix}{4-digit}/PRJ-{project-serial}/DGP-{group#}/{user} format**
- **Instruction IDs: INN-{prefix}{4-digit}/kwb-{kb-serial}/PRJ-{project-serial}/DGP-{group#}/{user} format**
- **File IDs inherit parent chat serial numbers correctly**
- **Collision prevention across all developer groups (1, 2, 3)**

Integration Testing:

- **Oracle NoSQL integration working**
- **Technical MCP integration for project chats and KB operations**
- **Direct n8n integration for group chats**
- **File upload and processing workflow**
- **Chat history creation and retrieval**
- **Performance targets met for all API endpoints**

Business Logic Testing:

- **Group chat creation and messaging (NO MCP)**
- **Project creation and management**
- **KB upload and processing (additive)**
- **Instruction CRUD operations (one per KB rule)**
- **Project chat creation with KB selection (MCP CRITICAL)**

- **File uploads with proper linkage to parent chats**
- **Chat history separation (group vs project)**
- **Error handling for rule violations**

Summary

This updated Green Services documentation provides complete content processing functionality for the Paradigm App build and test environment. The system uses proper system-generated UUIDs, supports the dual chat architecture (Group Chat + Project Chat), and includes comprehensive instruction management.

Key Features:

- ✅ **30 Green Services APIs with proper system UUID integration**
- ✅ **Group Chat (direct n8n) vs Project Chat (via Technical MCP) separation**
- ✅ **Complete knowledge base and instruction management**
- ✅ **File upload with serial number inheritance from parent chats**
- ✅ **Chat history management (separate for group and project chats)**
- ✅ **Collision-free UUID system across all developer groups**
- ✅ **MCP routing for AI-enhanced features**
- ✅ **Perfect integration with Black Services authentication**

Ready for Implementation:

- **Junior backend developers can build all 30 Green Services APIs**
- **Complete testing framework ensures quality**
- **Oracle NoSQL database integration**
- **Technical MCP and n8n workflow integration**

Testing Checklist

- **All 29 APIs respond within target times**
- **UUID generation follows patterns**
- **KB selection rules enforced**
- **File upload UIDs generated correctly**
- **MCP routing works for critical APIs**
- **Direct n8n routing works for standalone**
- **Chat isolation maintained**
- **Oracle NoSQL integration working**