Daily Work Summary - February 6, 2026

Project: Britannica Overseas Agent CRM
Developer: Ritik Saini
Module: Authentication & Authorization System

🎯 Core Objective

Implemented a **unified role-based password recovery system** supporting multiple user types (Admin, Agent, Student) with intelligent routing, enhanced security, and seamless UX.

🔧 Technical Implementation

1. Backend Architecture Refactoring

File Modified: `backend/src/controllers/authController.js`

A. Forgot Password Flow (forgotPassword function)

Problem Solved:

- Previously had separate endpoints for different user types
- Student password reset was broken
- No unified approach for multi-role authentication

Solution Implemented:

- Created unified `/auth/forgot-password` endpoint accepting `role` parameter
- Implemented **dynamic table routing pattern**:

```
if (role === 'STUDENT') → Query Student table
if (role === 'AGENT') → Query Agent table
if (role === 'ADMIN' || 'SUPER_ADMIN') → Query User table
```

- Added **multi-layer validation**:
  - Email validation
  - Role validation

    - Student completion status check (`isCompleted: true`)
    - Password setup status check
- Implemented **conditional logic** for password setup vs OTP:
    - If password not set → Send setup link (24hr token)
    - If password exists → Generate & send OTP (10min validity)
- Enhanced with **retry mechanism** for email delivery (3 attempts with exponential backoff)

Technical Highlights:

- Single endpoint replaces 3 separate endpoints (code reduction ~40%)
- Role-based polymorphic database queries
- Fallback email delivery system
- Comprehensive error logging

B. OTP Verification Flow (verifyOTP function)

Implementation:

- Accepts `role` parameter for correct table routing
- Validates OTP and expiration timestamp (`passwordResetOTPExpires`)
- Generates secure reset token (1-hour validity)
- Clears OTP after successful verification (security best practice)
- Role-aware response handling

Security Features:

- Time-based OTP expiration (10 minutes)
- Token-based password reset (1 hour)
- Automatic cleanup of used OTPs
- Database-level timestamp validation

---

2. Frontend Integration & State Management

Files Modified: 4 components

A. Login.jsx

- **Feature:** Automatic role parameter forwarding
- **Implementation:**
    - Added `useSearchParams` hook to extract role from URL
    - Dynamic forgot password link: `/forgot-password?role=${formData.role}`
    - Safety guard: Redirect to home if no role parameter

- **Impact:** Eliminates user confusion, maintains context

B. ForgotPassword.jsx

- **Complete Rewrite:** 180+ lines of logic
- **Key Features:**
  1. **Smart Role Selection:**
     - Reads role from URL query parameter
     - Shows dropdown only when role missing
     - Role options: Admin, Agent, Student (excludes SuperAdmin)
  2. **Conditional Rendering:**
     - Role from URL → Direct to email input
     - No role → Force selection first
  3. **API Integration:**
     - Single endpoint: `POST /auth/forgot-password`
     - Payload: `{ email, role }`
  4. **Navigation Logic:**
     - Success → `/verify-otp?role=${role}&email=${email}`
     - Back → `/login?role=${role}`

C. VerifyOTP.jsx

- **Enhanced Features:**
  - Role extraction from URL parameters
  - OTP input with auto-formatting (6-digit validation)
  - Countdown timer for resend (60 seconds)
  - Loading states during verification
- **API Calls:**
  - Verify: `POST /auth/verify-otp { email, otp, role }`
  - Resend: `POST /auth/forgot-password { email, role }`
- **Navigation:**
  - Success → `/reset-password?token=${resetToken}&role=${role}`
  - Back → `/forgot-password?role=${role}`

D. ResetPassword.jsx

- **Password Validation:**
  - Real-time strength indicator
  - Requirements checklist (length, uppercase, lowercase, numbers, special chars)
  - Match confirmation
- **Security:**

    - Token validation from URL
    - Role-preserved redirect after success
- **UX:**
    - Show/hide password toggle
    - 3-second auto-redirect with countdown
    - Manual "Go to Login" button

🔐 Security Enhancements Implemented

1. **Multi-Factor Validation:**
    - Email existence check
    - Role authorization
    - Student completion status (`isCompleted`)
    - Token/OTP expiration
2. **Time-Based Security:**
    - OTP: 10-minute validity
    - Reset Token: 1-hour validity
    - Password Setup Token: 24-hour validity
3. **Data Protection:**
    - Silent fail for non-existent emails (prevents user enumeration)
    - Automatic OTP cleanup after use
    - Secure token generation (crypto.randomBytes)
4. **Audit Trail:**
    - Comprehensive logging of all password reset attempts
    - User ID, email, role, and table name tracking
    - Failed attempt monitoring

🔄 Complete Integration Flow

```
┌──────────────────────────────────────────────────────────┐
│  Login Page (/login?role=STUDENT)                        │
│  ↓ Click "Forgot Password"                               │
├──────────────────────────────────────────────────────────┤
│  Forgot Password (/forgot-password?role=STUDENT)         │
```

```
│  ↓ Enter email → POST /auth/forgot-password                  │
│  ↓ Backend: Query Student table → Generate OTP               │
├──────────────────────────────────────────────────────────────┤
│  Verify OTP (/verify-otp?role=STUDENT&email=xxx)             │
│  ↓ Enter OTP → POST /auth/verify-otp                         │
│  ↓ Backend: Validate OTP → Generate reset token              │
├──────────────────────────────────────────────────────────────┤
│  Reset Password (/reset-password?token=xxx&role=...)         │
│  ↓ Enter new password → POST /auth/reset-password            │
│  ↓ Backend: Validate token → Update password                 │
├──────────────────────────────────────────────────────────────┤
│  Success → Redirect to Login (/login?role=STUDENT)           │
└──────────────────────────────────────────────────────────────┘
```

Role parameter preserved at every step (query string propagation)

📊 Code Metrics

Metric	Count
Files Modified	5 (1 backend, 4 frontend)
Functions Updated	2 (forgotPassword, verifyOTP)
Lines of Code	~300+ lines modified/added
API Endpoints Refactored	2
Database Tables Integrated	3 (User, Agent, Student)
Security Validations Added	8+ checkpoints

🐛 Issues Resolved

1. **Bug:** Student forgot password non-functional
    - **Root Cause:** Missing endpoint for Student table
    - **Fix:** Unified endpoint with role-based routing
2. **Bug:** Role context lost during navigation
    - **Root Cause:** URL parameters not preserved
    - **Fix:** Query string propagation across all redirects

3. **Bug:** OTP verification failing for students
    - **Root Cause:** Missing `isCompleted` check
    - **Fix:** Added completion validation in verifyOTP
4. **Bug:** Inconsistent redirect behavior
    - **Root Cause:** Hardcoded URLs without role parameter
    - **Fix:** Dynamic URL construction with role preservation

---