**Project-Based Coding Test: OSCE Scheduling System**

**Objective:**

Develop a drag-and-drop enabled web application that:

* Imports examinee, examiner, and standardized client data from uploaded Excel files.
* Automatically generates a schedule based on defined assignment and rotation conditions.
* Displays and manages tracks/stations via a drag-and-drop interface.

**Project Structure**

**Frontend (AngularJS + PrimeNG)**

* Dashboard to upload Excel files.
* Interface to configure number of administrations, tracks, and stations.
* Drag-and-drop UI to manually adjust station assignments per track.
* Button to trigger schedule generation.
* Table output for rotation schedule (Examinee × Station × Round).

**Backend (Node.js + Express + MongoDB)**

* REST API endpoints:
* POST /upload: Accept Excel files.
* POST /generate-schedule: Accept configurations and generate schedule.
* GET /schedule/:examId: Fetch generated schedule.
* Business logic to validate and assign users per constraints C1–C14.
* Mongoose schema for each entity (examinee, examiner, client, administration, track, station, rotation).

**Developer Tasks**

**Frontend Developer (AngularJS & PrimeNG)**

1. File Upload UI:
   * Accept .xlsx files for examiners, standardized clients, and examinees.
   * Show upload progress and validation status.
2. Admin Configuration Interface:
   * Add/remove administrations, tracks, and stations.
   * Assign tracks to administrations.
   * Assign stations to tracks using drag-and-drop.
3. Rotation Schedule Viewer:
   * Display resulting rotation schedule in a grid format.
   * Enable export to Excel or CSV.

Suggested Tech to be used:

* angular-file-upload or similar for file handling.
* PrimeNG components: p-table, p-fileUpload, p-dragDrop.

**Backend Developer (Node.js & MongoDB)**

1. Input Validation Module:
   1. Validate Excel structure (C14).
   2. Check availability of examiners/clients (C13).
2. Scheduling Algorithm Implementation:
   1. Implement logic for examinee assignment (C1–C5).
   2. Enforce examiner/client station rules (C6–C9).
   3. Ensure correct rotation logic (C10–C12).
   4. Raise exceptions or warnings on constraint violations.

**Data Models:**

Examinee

Examiner

Client

Station

Track

Administration

RotationSlot

**Export Functionality:**

Convert rotation schedule to Excel using xlsx or exceljs.

**Assessment Criteria**

**Category Weight Key Skills Evaluated**

Data Model & API Design 20% REST principles, schema design

Constraint-Based Scheduling Logic 30% Algorithm & edge-case handling

UI/UX Implementation (Drag/Drop) 20% AngularJS, PrimeNG, event handling

Data Validation & Error Handling 15% Input sanitization, user feedback

Code Quality & Documentation 15% Readability, modularity, comments

**Sample Files Provided to Candidates**

examinees.xlsx

examiners.xlsx

standardized\_clients.xlsx

admin\_template.json (template for number of tracks and stations per administration)