Technical Specification — Interviewer Calendar System

1. Project Overview

Build a web-based Interviewer Calendar System that lets interviewers invite candidates, collect candidates' availability, and schedule interviews either manually or automatically. The system should look and behave like a normal calendar for ease of use, include admin rights for the owner account, and provide a scheduling algorithm that produces optimal interview times covering the maximum number of candidates. The system must notify the interviewer of unavoidable conflicts and surface them in the UI for manual resolution.

2. Primary Actors

- Owner / Admin — the main account owner with full admin rights.
- Interviewer — can invite candidates, view aggregated candidate availabilities, and schedule interviews.
- Candidate — registers via Google OAuth and enters availability windows.

3. Key User Flows

- Invite candidate via email → secure registration link.
- Candidate registers via Google OAuth → enters availability in calendar.
- Interviewer views aggregated availabilities.
- Interviewer schedules interviews manually or via algorithm.
- Conflicts are flagged in the UI for manual resolution.

4. Functional Requirements

- Authentication & Authorization (Google OAuth, role-based access).
- Invitations with secure expiring tokens.
- Calendar & Availability management.
- Scheduling (manual and automatic).
- Conflict handling & notifications.

5. Scheduling Algorithm

- MVP: BFS-based search over time slots.
- Optimal: Maximum Bipartite Matching / Max-Flow.
- Conflict detection & explanation: identify unschedulable candidates.

6. Data Model (high level)

- Users: id, name, email, role, timezone, OAuth id.
- Invitations: inviter_id, invitee_email, token, expiry.
- Availabilities: candidate_id, start_datetime, end_datetime.
- Interviews: interviewer_id, candidate_id, start/end, status.
- Settings: working hours, slot granularity, notification templates.

7. API Endpoints (examples)

- POST /api/invitations — create/send invitation.
- POST /api/auth/google — handle OAuth login.
- GET /api/candidates/{id}/availabilities — list availabilities.
- POST /api/interviewer/{id}/schedule/manual — manual booking.
- POST /api/interviewer/{id}/schedule/automatic — automatic scheduling.

8. UI/UX Requirements

- Clean calendar UI (month/week/day views).
- Candidate availability editor.
- Scheduler modal (duration, buffer, date range).
- Conflict resolution panel.

9. Non-functional Requirements

- Timezone-aware (store UTC, display local).
- Security (OAuth best practices, secure tokens).
- Scalability (thousands of candidates).
- Reliability (idempotent operations, audit logs).
- Accessibility compliance.

10. Acceptance Criteria & Edge Cases

- Candidate with single short slot → flagged as conflict.
- Change in candidate availability triggers notification.
- Multiple interviewers scoped correctly.
- Automatic scheduler maximizes coverage.
- Conflict sets explained in UI.

11. Deliverables (MVP)

- OAuth login & role management.
- Invitation workflow.
- Availability input & persistence.
- Interviewer calendar (aggregated view).
- Manual and automatic scheduling.
- Conflict reporting & notifications.
- Unit tests for scheduler.
- Deployment docs.

12. Stretch Goals

- Recurring availability rules.
- Google Calendar / Outlook sync.
- Multi-interviewer scheduling.
- Candidate self-service rescheduling.
- Analytics reporting.

13. Acceptance Scenario (example)

Interviewer invites 10 candidates. 8 get scheduled automatically, 2 conflict. System flags conflicts and suggests manual resolution.