**Lean Platform Technologies**

Backend Developer Intern - Task Assignment

Spring Boot project with Java as the backend. We'll create RESTful APIs to handle the functionalities specified. For the sake of simplicity, we'll use an in-memory database (H2) and focus on implementing the core functionalities as per the provided specifications.

Below is a basic structure of how the backend code might look like:

// ConsultantRequest.java

public class ConsultantRequest {

private String id;

private String name;

private String email;

// Other fields and getters/setters

}

// Consultant.java

public class Consultant {

private String id;

private String name;

private int age;

private String phoneNo;

private String cv;

// Other fields and getters/setters

}

// WebAppStatistics.java

public class WebAppStatistics {

private int totalConsultants;

private int totalClients;

private int totalSessionsBooked;

// Getters/setters

}

// ConsultantService.java

public interface ConsultantService {

List<ConsultantRequest> getAllConsultantRequests();

ConsultantRequest approveConsultantRequest(String requestId);

ConsultantRequest rejectConsultantRequest(String requestId);

Consultant addConsultantDetails(Consultant consultant);

List<Consultant> getAllConsultants();

Consultant getConsultantByName(String name);

Consultant getConsultantByJobRole(String jobRole);

WebAppStatistics getWebAppStatistics();

}

// ConsultantServiceImpl.java

@Service

public class ConsultantServiceImpl implements ConsultantService {

// Implementation of all methods defined in ConsultantService

}

// ConsultantController.java

@RestController

@RequestMapping("/api")

public class ConsultantController {

private final ConsultantService consultantService;

public ConsultantController(ConsultantService consultantService) {

this.consultantService = consultantService;

}

@GetMapping("/consultant-requests")

public List<ConsultantRequest> getAllConsultantRequests() {

return consultantService.getAllConsultantRequests();

}

@PostMapping("/consultant-requests/{requestId}/approve")

public ResponseEntity<ConsultantRequest> approveConsultantRequest(@PathVariable String requestId) {

ConsultantRequest approvedRequest = consultantService.approveConsultantRequest(requestId);

return ResponseEntity.ok(approvedRequest);

}

@PostMapping("/consultant-requests/{requestId}/reject")

public ResponseEntity<ConsultantRequest> rejectConsultantRequest(@PathVariable String requestId) {

ConsultantRequest rejectedRequest = consultantService.rejectConsultantRequest(requestId);

return ResponseEntity.ok(rejectedRequest);

}

@PostMapping("/consultants")

public ResponseEntity<Consultant> addConsultantDetails(@RequestBody Consultant consultant) {

Consultant addedConsultant = consultantService.addConsultantDetails(consultant);

return ResponseEntity.ok(addedConsultant);

}

// Implement other endpoints for getting consultants, statistics, etc.

}

In this code:

ConsultantRequest, Consultant, and WebAppStatistics are POJO classes representing the entities involved.

ConsultantService defines the service contract.

ConsultantServiceImpl implements the service logic.

ConsultantController defines the REST endpoints to interact with the services.