# **Summary**

### **Website Summary: Employee Shift Calendar Solution**

The website is a modern, user-friendly platform designed to simplify and streamline shift scheduling for businesses with shift-based employees. It supports small, medium, and large enterprises and offers tailored experiences for three types of users: regular employees, team leaders, and administrators.

### **Key Features**

1. **Personalized Shift Management**:
   * Employees can view their weekly or monthly schedules with detailed information about each shift, including times, locations, and breaks.
   * Shifts are displayed in a visually clear calendar format, making it easy to understand work hours and availability.
2. **Shift Preferences**:
   * Employees can set their preferred working hours or mark times they are unavailable.
   * Preferences can be set to recurring and help administrators consider employee needs when planning shifts.
3. **Shift Swap and Requests**:
   * Employees can request to swap shifts with eligible colleagues or submit a change request directly to administrators.
   * The system ensures all requests meet company rules and labor regulations before approval.
4. **Team View for Managers**:
   * Team leaders can view all team members’ schedules in a simplified calendar.
   * Filters allow managers to quickly find specific employees or shifts based on team, role, or location.
5. **Overtime and Time of Management**:
   * Employees can view and manage their overtime requests, time off and allowances in one place.
   * Administrators can approve or reject overtime requests based on business needs.
6. **Comprehensive Admin Tools**:
   * Administrators can create, assign, and modify shifts for all employees.
   * The system ensures compliance with company policies and labor laws, such as minimum rest time between shifts and maximum working hours or minimum working hours.
   * Admins and to a less degree, team leaders, can approve shift swaps and manage employee profiles, roles, and shift eligibility.

### **Why It’s Valuable**

* **For Employees**: Provides transparency and flexibility by allowing them to view schedules, set preferences, and request changes directly through an intuitive interface.
* **For Team Leaders**: Simplifies team management by offering a clear view of everyone’s schedules and enabling swift decision-making.
* **For Administrators**: Reduces scheduling complexity with tools to create compliant and optimized schedules while accommodating employee needs.

### **How It Works**

* Accessible from any device, the website allows users to log in securely and access their personalized dashboard.
* Employees and managers interact through dynamic calendars and forms to manage shifts and requests.
* Administrators handle approvals and scheduling through robust management tools, ensuring operational efficiency and fairness.

This platform empowers employees, managers, and administrators to collaborate effectively, improving workforce satisfaction and productivity while maintaining compliance with company policies and labor laws. It’s a complete solution for shift scheduling in modern workplaces.

4o

## **Project Requirements with Tech Stack Integration**

### **Functional Requirements with Tech Stack Mapping**

### **General Features**

1. **User Authentication & Role Management**:
   * **Backend**: Implement authentication using Node.js with Express.js and bcrypt for password hashing. Use JWT for secure session management.
   * **Database**: Use PostgreSQL to store user details, including roles (Agent, Team Lead, Admin).
   * **Frontend**: React with vite will provide a secure login interface with role-specific dashboard redirection.
   * **Hosting**:
     + Frontend hosted on **Vercel**.
     + Backend hosted on **Heroku**, with SSL for secure API calls.
2. **Accessibility Features**:
   * **Frontend**: Use React with accessibility tools like react-aria for screen reader compatibility.
   * **Deployment**: Ensure consistent accessibility testing before deploying on **Vercel**.

### **Agent Section**

1. **Schedule Tab**:
   * **Frontend**:
     + Use a React calendar library (react-big-calendar or fullcalendar-react) to implement daily/weekly shift views.
     + Display shifts with details like type, location, and breaks using dynamic API data.
   * **Backend**:
     + Provide shift data via APIs (e.g., /api/shifts) built with Express.js.
     + Include business logic for time zone conversion and labor regulation compliance.
   * **Database**:
     + Store shifts in PostgreSQL, including fields for type, location, break times, and working hours.
   * **Hosting**:
     + Frontend connects to Heroku-hosted APIs over HTTPS.
2. **Team Schedule Tab**:
   * **Frontend**:
     + Build a table/grid view using React to show team schedules, with filters for agent attributes.
     + Add interactive filters for team, role, and department.
   * **Backend**:
     + Provide filterable team schedule data via APIs (e.g., /api/team-schedules).
   * **Database**:
     + Create relationships in PostgreSQL to link users with teams and roles.
3. **Preferences Tab**:
   * **Frontend**:
     + Use React forms to create/modify preferences.
     + Implement recurring preferences logic (e.g., React Date Picker for dates).
   * **Backend**:
     + APIs for creating, updating, and deleting preferences (e.g., /api/preferences).
     + Validate preferences to ensure compliance with shift rules.
   * **Database**:
     + Store preference rules (e.g., unavailable times, preferred hours) in PostgreSQL.
4. **Overtime Tab**:
   * **Frontend**:
     + Build an overtime request table in React, showing pending and approved requests.
   * **Backend**:
     + Manage overtime requests via APIs (e.g., /api/overtime).
   * **Database**:
     + Store overtime requests in PostgreSQL with status (pending/approved/rejected).
5. **Swaps Tab**: (request)
   * **Frontend**:
     + Implement a swap request form using React, with validation for eligibility and constraints.
   * **Backend**:
     + APIs for shift swap requests (e.g., /api/swaps).
     + Validate swaps based on agent eligibility and labor regulations.
   * **Database**:
     + Maintain swap requests in PostgreSQL with agent details and approval statuses.

### **Admin Section**

1. **Shift Allocation**:
   * **Backend**:
     + API for allocating shifts to agents (e.g., /api/admin/shifts).
     + Validate shifts against rules like eligibility and labor laws.
   * **Database**:
     + Store allocation data in PostgreSQL with shift constraints.
   * **Frontend**:
     + Admin dashboard built with React to manage shift assignments interactively.
2. **Pending Swaps**:
   * **Backend**:
     + Manage pending swap requests via APIs (e.g., /api/admin/swaps).
     + Include logic for multi-level approvals (Team Lead, Admin).
   * **Database**:
     + Record approval workflows in PostgreSQL.
   * **Frontend**:
     + Admin view in React with sortable lists of swap requests.
3. **Manage Agents**:
   * **Backend**:
     + APIs for CRUD operations on agent profiles (e.g., /api/admin/agents).
   * **Database**:
     + Maintain agent attributes (e.g., team, location, shift eligibility) in PostgreSQL.
   * **Frontend**:
     + Admin panel in React to modify agent data interactively.
4. **Additional Requirements**:
   * **Backend**:
     + API to add extra instructions to shifts (e.g., /api/admin/requirements).
   * **Database**:
     + Add a requirements column to the shifts table in PostgreSQL.
   * **Frontend**:
     + React modal for admin to add additional requirements to shifts.
5. **Compliance Checks**:
   * **Backend**:
     + Validate shifts and preferences using country-specific labor law rules (e.g., minimum time between shifts).
   * **Database**:
     + Store labor law configurations for different locations in PostgreSQL.

### **Non-Functional Requirements with Tech Stack**

1. **Performance**:
   * PostgreSQL for optimized database queries.
   * Use query indexing and caching mechanisms (e.g., Redis for caching).
2. **Scalability**:
   * Use Heroku’s free tier with autoscaling for backend and database.
   * React’s component-based structure ensures scalability on the frontend.
3. **Security**:
   * Backend hosted on Heroku with enforced SSL.
   * HTTPS for API communication.
   * Use helmet in Node.js for secure headers and cors for domain restrictions.
   * Encrypt sensitive data in PostgreSQL (e.g., user passwords with bcrypt).
4. **Deployment**:
   * **Frontend**:
     + Continuous deployment on Vercel, triggered by GitHub pushes to main.
   * **Backend**:
     + Automatic deployment on Heroku with GitHub integration.
   * **Database**:
     + Heroku PostgreSQL Free Tier with SSL-enabled connections.

### **Integration Workflow**

1. **Frontend**:
   * Connect React components to the backend using Axios or Fetch.
   * Use .env files for environment-specific API URLs.
2. **Backend**:
   * Design RESTful APIs for all functionalities.
   * Use middleware for validation and error handling.
3. **Database**:
   * Design schema with foreign key relationships.
   * Use migrations for schema changes.
4. **Deployment**:
   * Configure CI/CD pipelines for both Vercel (frontend) and Heroku (backend).
   * Use environment variables in both platforms for secure configuration.

### **Tech Stack Mapping to Requirements Summary**

| **Requirement** | **Frontend (React)** | **Backend (Node/Express)** | **Database (PostgreSQL)** | **Hosting** |
| --- | --- | --- | --- | --- |
| User Authentication | Login/Register Components | Auth API, JWT handling | Users Table | Vercel (FE), Heroku (BE) |
| Schedule Management | Calendar UI | Shift APIs | Shifts Table | Vercel (FE), Heroku (BE) |
| Preferences | Forms for Preferences | Preferences API | Preferences Table | Vercel (FE), Heroku (BE) |
| Swaps | Swap Request UI | Swap APIs | Swaps Table | Vercel (FE), Heroku (BE) |
| Admin Features | Admin Dashboard | Admin APIs | Various Tables (Agents, Shifts, Swaps) | Vercel (FE), Heroku (BE) |

T