**1. Appointment Booking System**

Components:

* **AppointmentBooking.js**: Main container that loads the appropriate sidebar based on user role (admin/employee) and renders the Appointment component.
* **Appointment.js**: Core appointment management interface with features to:
  + Book new appointments (employees)
  + View/accept/reject appointments (admins)
  + Cancel appointments (employees)
  + Filter and sort appointments
* **AppointmentTile.js**: Dashboard widget showing upcoming appointments

Backend Integration:

* **AppointmentController.cs**: Handles all appointment-related operations:
  + Status updates (Pending/Accepted/Rejected/Cancelled)
  + Validation of appointment times and dates
  + Role-based access control

**Workflow**:

1. User loads AppointmentBooking view, which checks their role
2. Appointment component fetches relevant appointments (all for admin, employee-specific for users)
3. Form submissions call API endpoints to create/update appointments
4. Status changes trigger updates in both the frontend and the database

**2. Payslip Management System**

Components:

* **PayslipManagement.js**: Admin interface for generating payslips
* **PayslipPage.js**: Employee view for accessing/downloading payslips
* **PayslipGenerator.js**: Core payslip generation tool with features to:
  + Generate individual or bulk payslips
  + Add bonuses/allowances
  + Preview payslip details

Backend Integration:

* **PayslipController.cs**: Handles payslip operations:
  + Generation of new payslips
  + PDF generation and download
  + Employee-specific payslip retrieval
  + Tax/UIF calculations via PayslipCalculator service
* **PayslipCalculator.cs**: Service that handles:
  + South African tax bracket calculations
  + UIF contribution calculations
  + Net salary computations

**Workflow**:

1. Admin uses PayslipGenerator to select employees/positions and generate payslips
2. System calculates deductions using PayslipCalculator
3. Payslip data is stored in the database, and PDFs are generated
4. Employees access their payslips through PayslipPage, which fetches their records

**3. Task Management System**

Components:

* **Todo.js**: Task management component with features to:
  + Add/edit/delete tasks
  + Mark tasks as complete
  + Filter by priority/status
  + Set due dates and reminders

Backend Integration:

* **TodoController.cs**: Handles task operations:
  + CRUD operations for tasks
  + Employee-specific task retrieval
  + Status updates
  + Due date validation

**Workflow**:

1. User loads their task list, which fetches employee-specific tasks
2. Tasks can be added/updated/deleted via API calls
3. Status changes are immediately reflected in the UI and database

**4. Document Management System**

Components:

* **Documents.js**: Main document management interface with features to:
  + Upload/download/view documents
  + Search/filter documents (admin only)
  + Delete documents (owner only)
* **DocumentsPage.js**: Wrapper component that integrates Documents with Sidebar
* **DocumentController.cs**: Backend controller for document operations

Backend Integration:

* **DocumentController.cs** handles:
  + File uploads with validation (type/size)
  + Secure file storage on server
  + Document metadata management in Supabase
  + Role-based access control
  + Employee search functionality

**Workflow**:

1. User accesses Documents page which checks their role
2. Documents component fetches relevant documents (all for admin, employee-specific for users)
3. Uploads are validated and stored with metadata
4. Downloads stream files directly from server storage
5. Admins can search documents by employee ID

**Key Features**:

* Secure file storage with unique filenames
* File type validation (PDF, DOC, JPG, etc.)
* Role-based access control
* Metadata tracking (upload date, size, type)
* Employee search for admins

**5. Leave Management System**

Components:

* **LeaveApplications.js**: Basic leave application UI (placeholder)
* **LeaveRequestAdmin.js**: Admin interface for managing leave requests
* **LeaveManagement.js**: Admin wrapper for leave management
* **LeaveRequestController.cs**: Backend controller for leave operations

Backend Integration:

* **LeaveRequestController.cs** handles:
  + Leave request submission
  + Status updates (Approved/Denied/Pending)
  + Employee-specific leave history
  + Validation of leave dates
  + Comprehensive logging

**Workflow**:

1. Employees submit leave requests via frontend
2. Admins review requests in LeaveRequestAdmin interface
3. Admins approve/deny requests which updates status
4. Employees can view their leave history
5. System validates leave dates and calculates total days

**Key Features**:

* Leave type tracking (sick, vacation, etc.)
* Status management with visual indicators
* Date validation to prevent invalid requests
* Comprehensive logging of all actions
* Employee-specific views

**6. Profile Management**

Components:

* **Profile.js**: User profile display component showing:
  + Personal information
  + Contact details
  + Employment information

**Key Features**:

* Dynamic sidebar rendering based on role
* Clean, organized presentation of user data
* Responsive design for all devices
* Secure data handling (no edit functionality)

**7. Landing Page**

Landing Page is where users will interact with the system for the first time

Components:

* RegisterForm.js – this is the form for when a user is registering, the form takes in the following fields:
* Name
* Surname
* Email
* Role
* ID Number
* Password
* Confirm Password
* LoginForm.js – Form used for logging in after successfully registering and being admitted. Fields include:
* Email
* Password
* Navbar.js – Houses company logo, only visible in the landing page

Pages:

* Landing.js

Backend Integration

* SupabaseTestController.cs – Handles validation that the fields are filled before registering and then registers the user
* LoginController.cs – Handles validating that the user is indeed in the database and admitted and then logs user I successfully if conditions are met

8. **Admin Dashboard**

If user role is an admin, they will be redirected to the admin dashboard.

Components:

* Dashboard.js – Contains summery of Tasks Admin can partake in
* AdminSidebar.js – All admin functions and navigation will be found here

9. **HR Admission**

This is where hr will be able to accept pending registration request. Once the pending registration requestion has been accepted, the employee will be automatically given an employee number and status changes to admitted.

Components:

* HRAdminForm.js – This is the contract popup which appears when hr clicks admit button. The point is to add contract details of the employee before officially admitting them into the system.
* AdminSidebar.js

Pages:

* EmployeeManagement.js – Handles employees management depending on their employment status. First, the non-admitted tab will show only pending users waiting to be admitted. Then the admitted tab for admitted users which shows all admitted users and gives you the option to offboard a user. If you successfully offboard a user, their record will be removed active users. Then there is offboarding tab which shows all offboarded users. Lastly, all users tab, will show all pending, admitted and offboarded users.

Backend Integration:

* HRAdminController.cs – Handles Admitting a user and changing their status from pending to admitted. It also handles offloading users from the users table and then adding them into the archived\_users table in the database. Fetches users based on their status.