PROJECT REPORT

SAILORSHIFT – SMART WORKFORCE SCHEDULER

Optimizing Workforce Management with Advanced Scheduling Solutions

Department of Mechanical, Industrial and Aerospace Engineering

Concordia University, Montreal, Canada

Faculty of Engineering and Computer Science

Professor Dr. Ali Akgunduz

INDU 6990 – Industrial Engineering Capstone

Winter 2025

April 10th, 2025

Submitted by:

Team 12

Sai Krishna Prashanth Kolluru (40277712)

Saisri Durga Shanmukha Abhiram Kalvakolanu (40277691)

Avinash Pothabattula (40288611)

Roshan Kotapati (40291859)

Lalith Chand Saripalli (40292708)

ABSTRACT

This project aims to develop a comprehensive and fully automated scheduling management system, capable of streamlining various critical tasks such as managing employee leave requests, facilitating shift swap requests, and generating user accounts automatically for large-scale adoption. The central objective was to create an intuitive, efficient, and fully integrated digital platform to handle complex administrative tasks effortlessly, minimizing manual intervention and significantly reducing managerial overhead. Throughout the project lifecycle, we successfully developed two distinct yet integrated web applications, one tailored specifically for employers and administrative staff, and another designed for employee interactions. The employer-facing website provides administrative capabilities to manage employee schedules, oversee leave and swap requests, and access analytical reports that facilitate informed decision-making. Conversely, the employee-facing website offers intuitive, user-friendly functionalities, enabling employees to easily request leaves, initiate shift swaps, view their schedules, and manage their profiles autonomously. Both platforms were rigorously tested in real-world scenarios through direct collaboration with local businesses, enabling practical feedback, iterative refinement, and feature enhancements based upon user experiences and requirements.

Although our original vision was designed with scalability for large-scale operations in mind, extensive testing revealed several challenges associated with mass deployment and scalability at enterprise levels, primarily relating to the complexity of integration with diverse, large-scale legacy systems. However, the developed system proved exceptionally beneficial and efficient for Small-to-Medium Enterprises (SMEs) and local businesses, demonstrating considerable value through significant administrative time savings, reduction of errors, and enhanced employee satisfaction due to transparent and efficient schedule management. Additionally, recognizing the increasing importance of seamless, real-time user support, we successfully integrated a conversational chatbot, leveraging the advanced Gemini 2.0 Flash AI framework. This intelligent chatbot effectively interprets natural-language queries from users, offering immediate assistance in processes such as leave request submission, shift swaps, account management queries, and general scheduling inquiries. The chatbot significantly enhances user experience by providing prompt responses, reducing support response times, and streamlining communication between employees and administration.

TABLE OF CONTENTS

1. INTRODUCTION

Efficient scheduling systems have become essential in modern organizations due to increasing operational complexities, workforce size, and dynamic business environments. Proper scheduling significantly influences organizational productivity, operational efficiency, employee satisfaction, and overall competitiveness. Without a structured scheduling system, businesses face challenges such as frequent errors in shift management, lack of transparency in leave approvals, inefficient handling of employee shift swaps, miscommunication among stakeholders, and increased administrative workload. These problems collectively diminish productivity and employee morale, resulting in lower profitability and competitiveness in the market.

Given these challenges, automation of scheduling systems has become a strategic necessity. Automated scheduling solutions help streamline complex organizational tasks such as managing employee availability, leave requests, shift allocation, and shift swaps. By automating these activities, businesses significantly reduce the potential for human error, save managerial time, enhance transparency, and ensure fairness in decision-making. Automation also improves accuracy and speed, providing employees and managers alike with real-time visibility into scheduling, leave approval statuses, and swap requests, resulting in enhanced communication and reduced conflict. Furthermore, automation allows organizations to allocate their human resources more strategically, shifting managerial focus from administrative tasks to more strategic, value-adding initiatives.

Employers and organizations benefit immensely from automated scheduling systems through improved efficiency, reduced administrative overhead, and enhanced decision-making capabilities derived from real-time data insights. The transparency brought by automation reduces conflicts and misunderstandings regarding scheduling decisions. Employees also benefit directly, experiencing increased flexibility, autonomy, and satisfaction. Employees can easily manage their leave requests, swap shifts seamlessly with peers, and access their schedules instantly, fostering a sense of control and involvement in their work-life management.

Recognizing the crucial impact scheduling optimization has on organizational efficiency, employee satisfaction, and overall operational performance, we, as industrial engineering students, selected this project as our capstone experience. Industrial engineering emphasizes improving organizational processes through efficient systems design, optimization techniques, and effective resource utilization. Thus, this project directly aligns with our academic discipline, applying theoretical knowledge of scheduling theory, optimization methods, database management systems, and artificial intelligence-driven interactions.

To successfully execute this project, we extensively leveraged scheduling theory to optimize workforce allocation, reduce inefficiencies, and enhance organizational productivity. Optimization models for leave and swap requests were integrated to ensure fair, equitable, and efficient resource distribution. Additionally, database management concepts were critical in creating robust backend storage and retrieval systems, ensuring data integrity, security, and seamless system performance. The implementation of a conversational AI chatbot using the Gemini 2.0 Flash AI platform provided a sophisticated front-end interaction, delivering instant assistance to users regarding scheduling issues, leave, and swap requests. To effectively integrate these diverse components, we acquired deep insights into scheduling theory, learned advanced optimization strategies, enhanced our proficiency in database management systems, and gained valuable experience in artificial intelligence and machine learning implementation.

The knowledge and skills obtained through this capstone project are highly valuable for our future careers. The experience of developing automated scheduling systems and integrating AI technologies provided us practical insights into real-world problem-solving, project management, interdisciplinary collaboration, and innovative thinking. In conclusion, by designing and developing this comprehensive automated scheduling solution, we addressed critical organizational challenges, enhancing both employee satisfaction and managerial effectiveness. The integration of AI-driven technology further enhanced the functionality, demonstrating our ability to apply industrial engineering principles practically and effectively. This project not only served as an important milestone in our academic journey but also equipped us with invaluable practical skills, paving the way for future contributions to organizations seeking similar operational excellence.

2. DISCUSSION

2.1. Web Application

The SailorShift web application consists of two main portals: an Employer Portal and an Employee Portal. Both portals are integrated with Supabase, a cloud-based PostgreSQL database, designed to automate workforce scheduling, manage leaves and shift swaps, and facilitate real-time interaction through a sophisticated AI chatbot.

Frontend: HTML, CSS, JavaScript, React Js, Node Js for interactive and responsive user interfaces.

Backend: Python Flask framework (app.py, func.py) ensuring smooth server operations, robust data handling, and chatbot interactions.

2.1.1. Employer Portal

The Employer Portal enables comprehensive management and monitoring of employees, scheduling tasks, and communication via an intuitive user interface.

1. Login Page

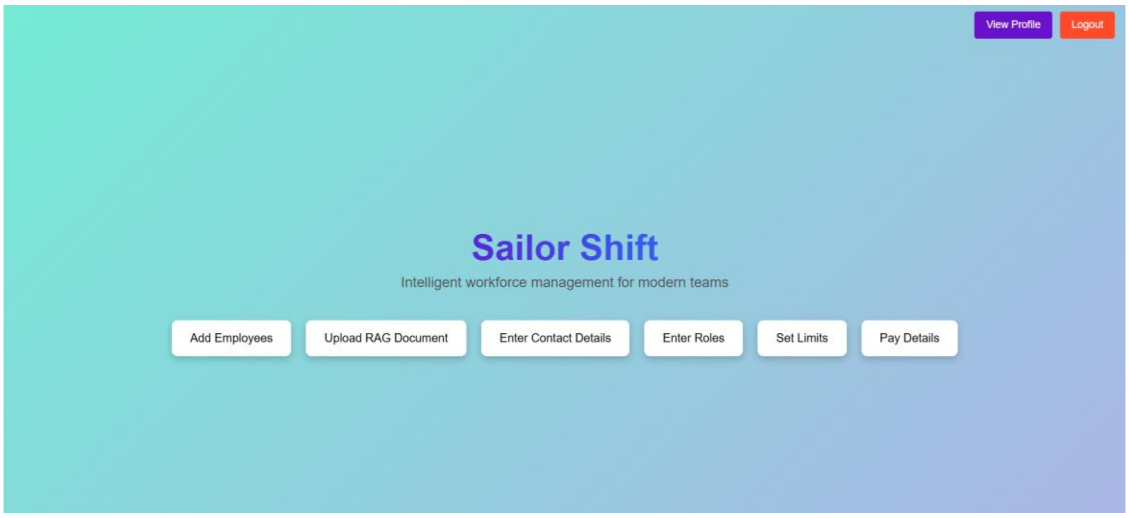
This page facilitates secure employer access to the portal. Employers input their username or email and password. First-time users must register their organization through the "Register your organization" page.

2. Register Your Organization Page

This page captures critical organizational details such as organization type, name, employer’s name, and email ID. After submission, login credentials are automatically sent to the employer's email, enabling immediate access to the portal.

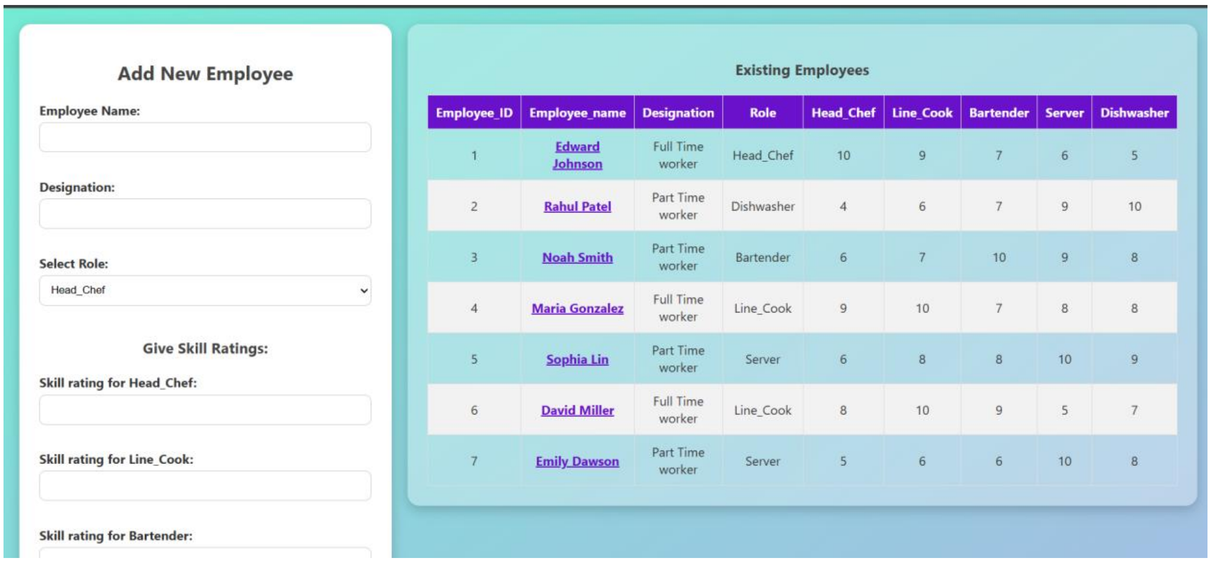
3. Home Page (Dashboard)

This comprehensive dashboard provides centralized control with various management functions, including employee addition, skill assignment, role definition, and employee credential management. The employer can also set operational hour limits and upload RAG documents for AI assistance. Seven main functionalities are accessible from this dashboard:



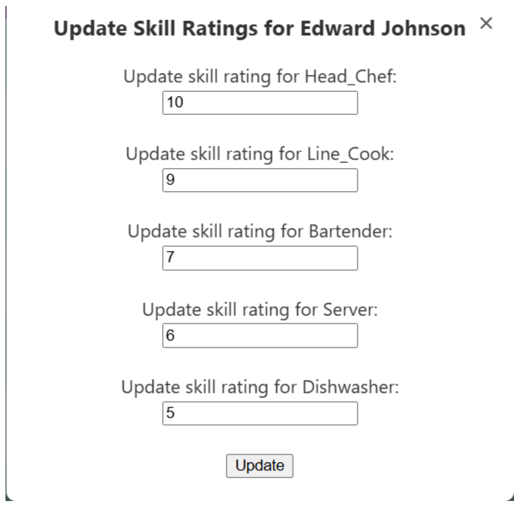
i. Add Employees

Employers can add new employees, assign roles, and input skill ratings. Skill ratings facilitate intelligent scheduling decisions by matching appropriate employees to relevant tasks. Skill ratings can also be updated dynamically through this page.



Existing Employees

Employee_ID	Employee_name	Designation	Role	Head_Chef	Line_Cook	Bartender	Server	Dishwasher
1	Edward Johnson	Full Time worker	Head_Chef	10	9	7	6	5
2	Rahul Patel	Part Time worker	Dishwasher	4	6	7	9	10
3	Noah Smith	Part Time worker	Bartender	6	7	10	9	8
4	Maria Gonzalez	Full Time worker	Line_Cook	9	10	7	8	8
5	Sophia Lin	Part Time worker	Server	6	8	8	10	9
6	David Miller	Full Time worker	Line_Cook	8	10	9	5	7
7	Emily Dawson	Part Time worker	Server	5	6	6	10	8

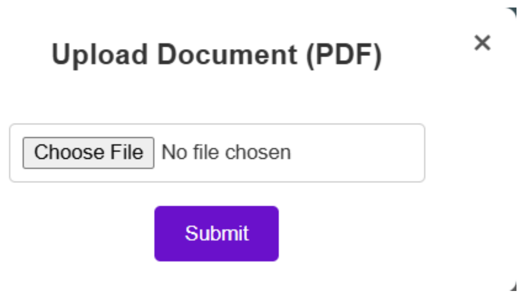


Update Skill Ratings for Edward Johnson

Update skill rating for Head_Chef:	10
Update skill rating for Line_Cook:	9
Update skill rating for Bartender:	7
Update skill rating for Server:	6
Update skill rating for Dishwasher:	5

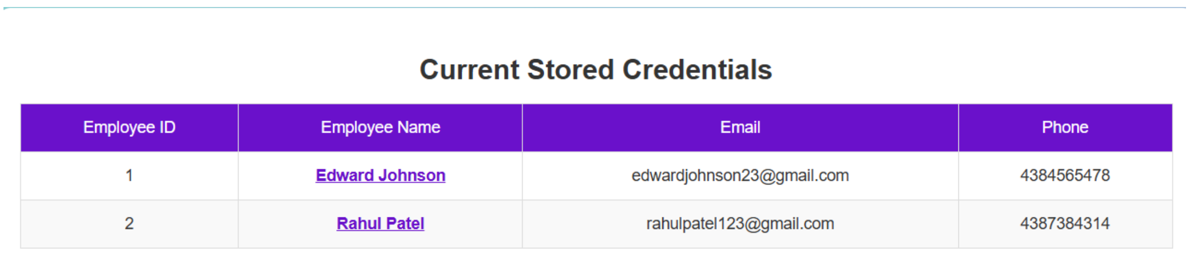
ii. Upload RAG Document

Employers upload PDF documents containing relevant organizational information. These documents empower the chatbot (through Retrieval-Augmented Generation or RAG) to answer queries posed by employees with high accuracy and context-awareness.



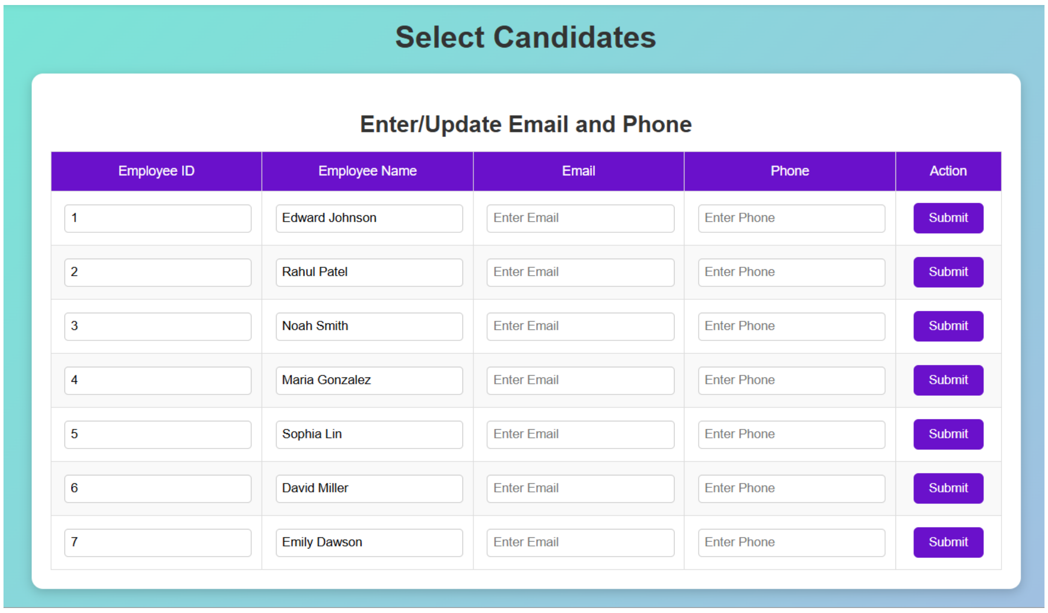
iii. Enter Contact Details

This page allows the employer to input or update employees' contact details, stored securely in the candidate_credentials Supabase table. Employers can also trigger automated email dispatch of employee login credentials, simplifying onboarding.



Current Stored Credentials

Employee ID	Employee Name	Email	Phone
1	Edward Johnson	edwardjohnson23@gmail.com	4384565478
2	Rahul Patel	rahulpatel123@gmail.com	4387384314

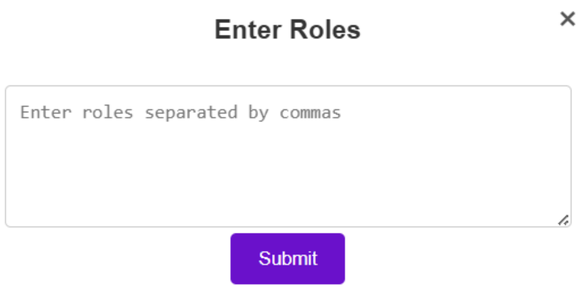

Select Candidates

Enter/Update Email and Phone

Employee ID	Employee Name	Email	Phone	Action
1	Edward Johnson	Enter Email	Enter Phone	Submit
2	Rahul Patel	Enter Email	Enter Phone	Submit
3	Noah Smith	Enter Email	Enter Phone	Submit
4	Maria Gonzalez	Enter Email	Enter Phone	Submit
5	Sophia Lin	Enter Email	Enter Phone	Submit
6	David Miller	Enter Email	Enter Phone	Submit
7	Emily Dawson	Enter Email	Enter Phone	Submit

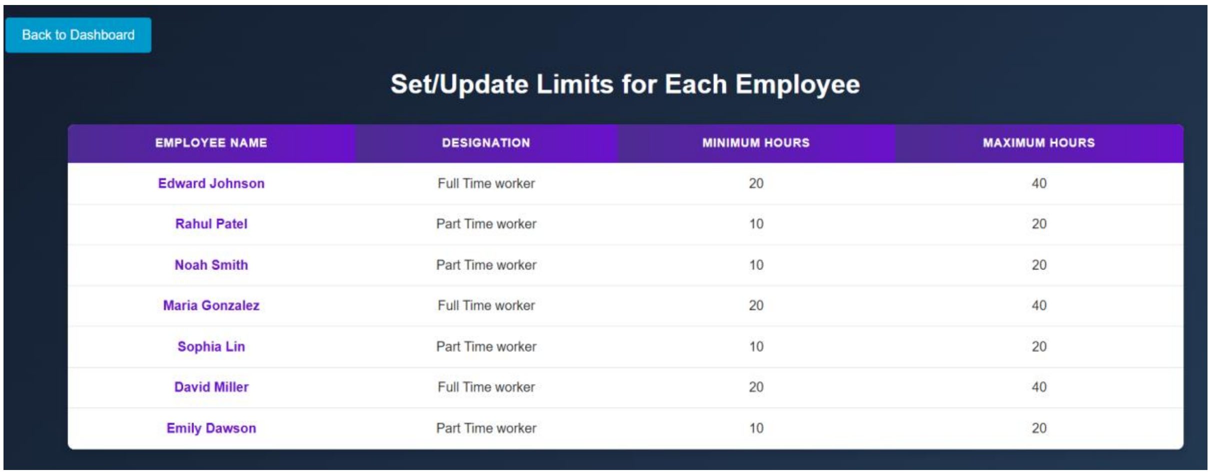
iv. Enter Roles

The initial setup involves defining organizational roles through a straightforward interface. Roles are essential for structuring employee assignments and are stored in the employee database table for reference in all scheduling decisions.


Enter Roles

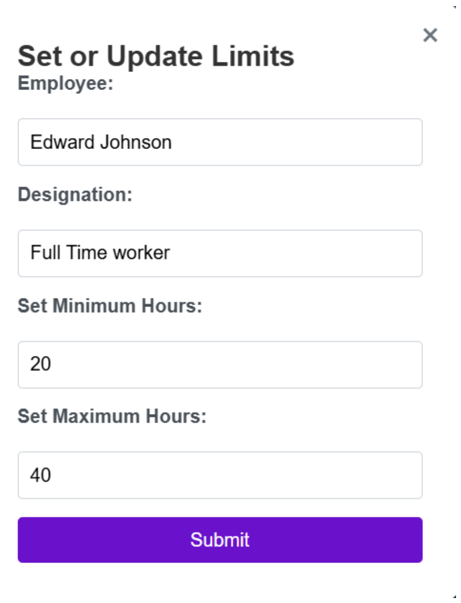
v. Set Limits

This functionality allows employers to set minimum and maximum hourly limits for each role, ensuring optimal staffing and regulatory compliance. These values are stored in the limits table and automatically considered in schedule creation.



Set/Update Limits for Each Employee

EMPLOYEE NAME	DESIGNATION	MINIMUM HOURS	MAXIMUM HOURS
Edward Johnson	Full Time worker	20	40
Rahul Patel	Part Time worker	10	20
Noah Smith	Part Time worker	10	20
Maria Gonzalez	Full Time worker	20	40
Sophia Lin	Part Time worker	10	20
David Miller	Full Time worker	20	40
Emily Dawson	Part Time worker	10	20



Set or Update Limits

Employee: Edward Johnson

Designation: Full Time worker

Set Minimum Hours: 20

Set Maximum Hours: 40

vi. Pay Details

A detailed table shows weekly schedules and corresponding pay calculations for each employee. Data is dynamically fetched from the changed_schedule table, incorporating all leave and swap adjustments. Automatic calculations of days worked and weekly pay enhance payroll accuracy and efficiency.

vii. View Profile

Employers can update organizational details and manage account settings, including password changes, enhancing profile security and account management convenience.

viii. Logout

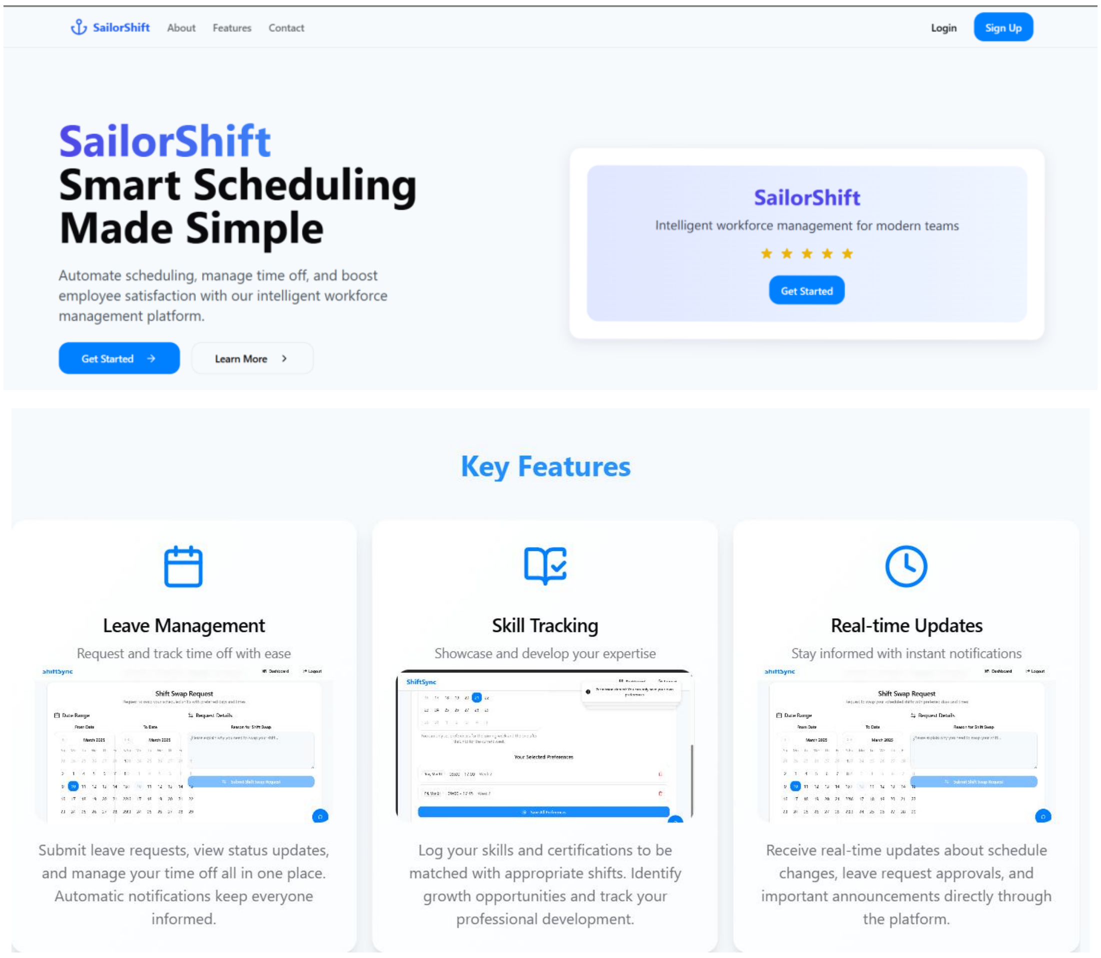
Secure logout functionality ensures data security, redirecting employers back to the login page.

2.1.2. Employee Portal

The Employee Portal offers a user-friendly interface allowing employees to manage schedules, preferences, and leave and shift swap requests effortlessly.

1. Home (Landing) Page

Serves as the entry point for employees, featuring introductory content and a clear call-to-action guiding users to login.



SailorShift
Smart Scheduling Made Simple

Automate scheduling, manage time off, and boost employee satisfaction with our intelligent workforce management platform.

SailorShift
Intelligent workforce management for modern teams

★ ★ ★ ★ ★

Key Features

Leave Management
Request and track time off with ease

Submit leave requests, view status updates, and manage your time off all in one place. Automatic notifications keep everyone informed.

Skill Tracking
Showcase and develop your expertise

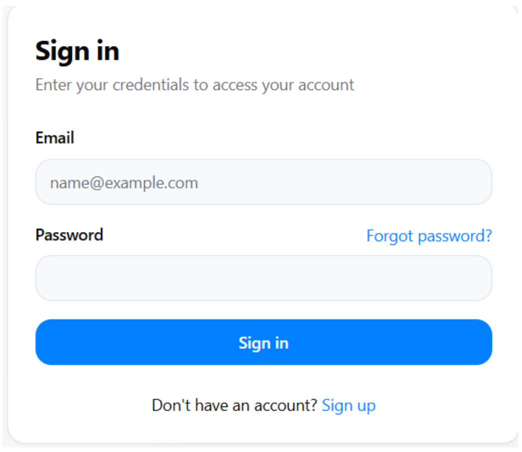
Log your skills and certifications to be matched with appropriate shifts. Identify growth opportunities and track your professional development.

Real-time Updates
Stay informed with instant notifications

Receive real-time updates about schedule changes, leave request approvals, and important announcements directly through the platform.

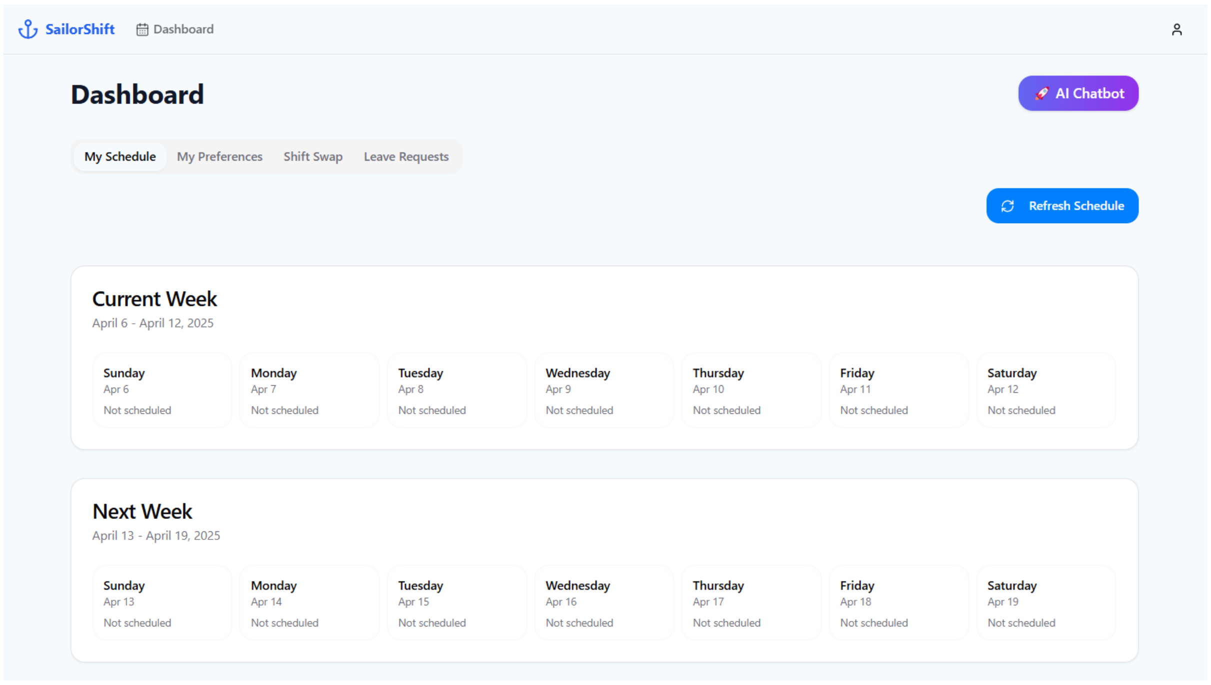
2. Get Started (Login) Page

Provides secure employee authentication, validating credentials against stored Supabase data (candidate_credentials), leading to personalized dashboard access.


Sign in
Enter your credentials to access your account
Email
name@example.com
Password
Forgot password?
Sign in
Don't have an account? Sign up

3. Dashboard

A central hub displaying personalized schedules and providing direct access to leave requests, shift swaps, and preferences management.



Current Week
April 6 - April 12, 2025

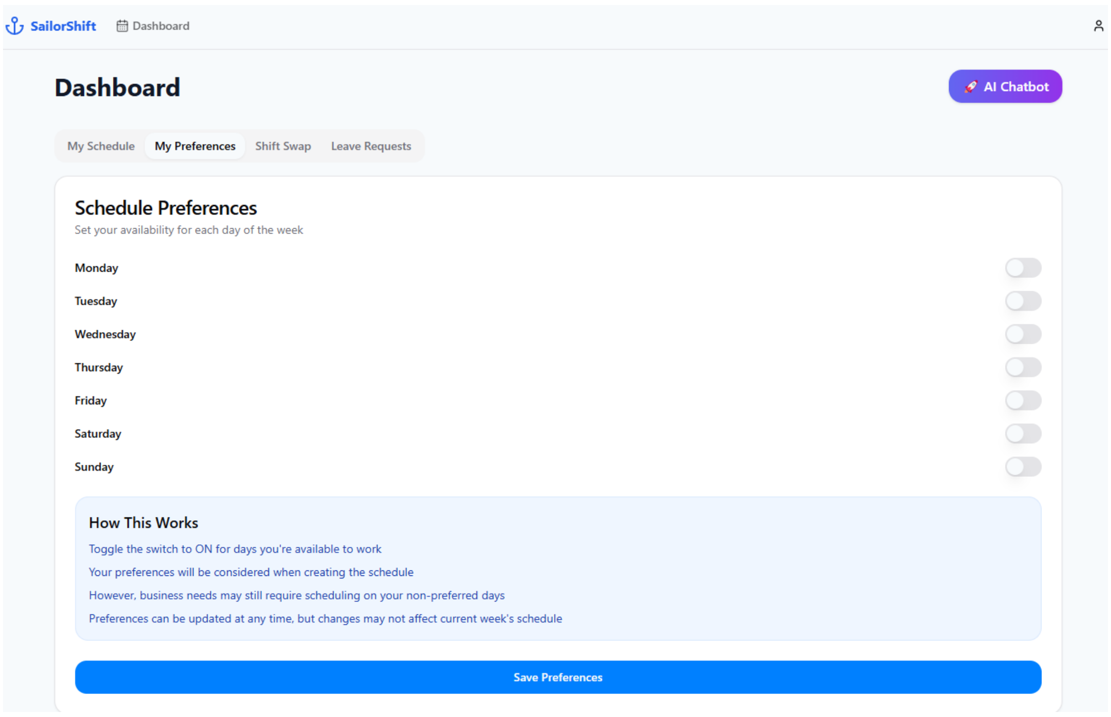
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Apr 6	Apr 7	Apr 8	Apr 9	Apr 10	Apr 11	Apr 12
Not scheduled	Not scheduled	Not scheduled	Not scheduled	Not scheduled	Not scheduled	Not scheduled

Next Week
April 13 - April 19, 2025

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Apr 13	Apr 14	Apr 15	Apr 16	Apr 17	Apr 18	Apr 19
Not scheduled	Not scheduled	Not scheduled	Not scheduled	Not scheduled	Not scheduled	Not scheduled

4. My Preferences Page

Employees select preferred working days and hours, which automatically update their schedules. Preferences are stored in the preferences table, aiding in dynamic scheduling and improved employee satisfaction.



Dashboard

My Schedule | My Preferences | Shift Swap | Leave Requests

Schedule Preferences

Set your availability for each day of the week

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday

Sunday

How This Works

Toggle the switch to ON for days you're available to work

Your preferences will be considered when creating the schedule

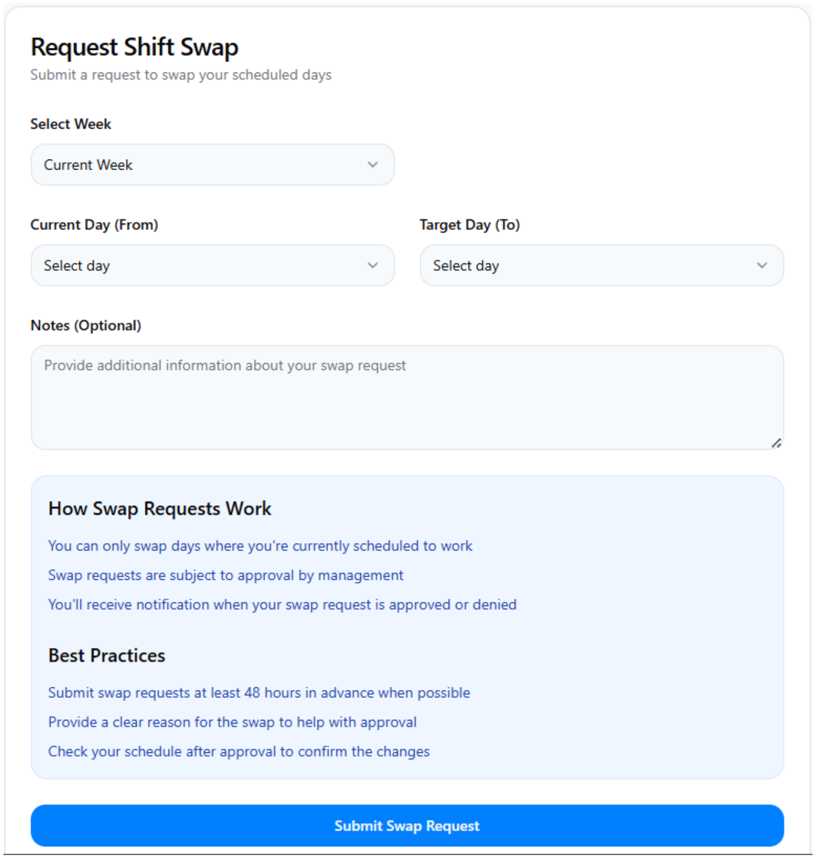
However, business needs may still require scheduling on your non-preferred days

Preferences can be updated at any time, but changes may not affect current week's schedule

Save Preferences

5. Shift Swap Page

Employees propose shift swaps and adjustments. Integrated validations prevent scheduling conflicts. Approved swaps are reflected instantly on the dashboard, maintaining real-time accuracy.



Request Shift Swap

Submit a request to swap your scheduled days

Select Week: Current Week

Current Day (From): Select day

Target Day (To): Select day

Notes (Optional): Provide additional information about your swap request

How Swap Requests Work

You can only swap days where you're currently scheduled to work

Swap requests are subject to approval by management

You'll receive notification when your swap request is approved or denied

Best Practices

Submit swap requests at least 48 hours in advance when possible

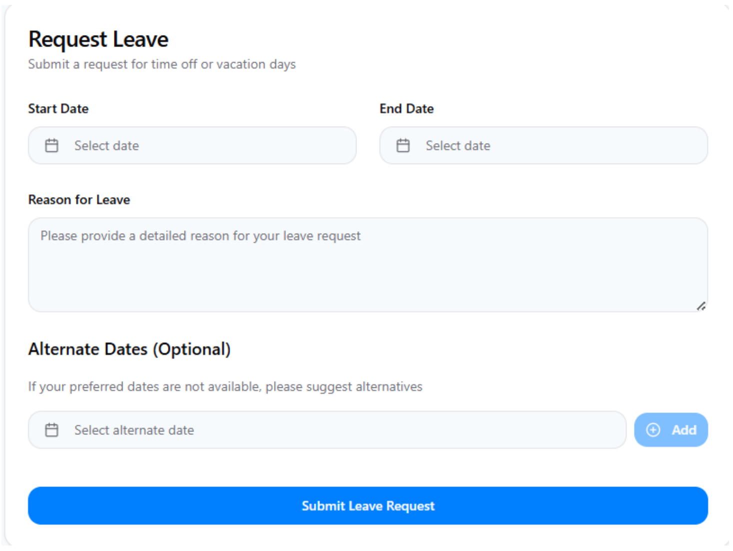
Provide a clear reason for the swap to help with approval

Check your schedule after approval to confirm the changes

Submit Swap Request

6. Leave Request Page

Enables easy submission and tracking of leave requests with integrated validation to ensure accuracy. Approved leave requests automatically update employee schedules stored in new_schedule and subsequently reflected in changed_schedule.



Request Leave

Submit a request for time off or vacation days

Start Date: Select date

End Date: Select date

Reason for Leave: Please provide a detailed reason for your leave request

Alternate Dates (Optional)

If your preferred dates are not available, please suggest alternatives

Select alternate date

2.2. Database Management (Cloud Supabase Integration)

Supabase and PostgreSQL efficiently store and manage all organizational data, providing seamless interaction between front-end and back-end operations. Key tables include:

1. employees: Employee data including roles and skill ratings.
2. candidate_credentials: Employee contact and login information.
3. limits: Operational hourly limits per role.
4. preferences: Employee scheduling preferences.
5. new_schedule: Initial schedule data prior to modifications.
6. changed_schedule: Dynamic schedule reflecting all changes from leave and swap activities.

2.3. AI Chatbot

2.3.1. Description of Chatbot

An AI-powered chatbot has been successfully developed and integrated into the organization’s system using a Flask-based Python application. This intelligent assistant is capable of handling a wide range of employee queries, including organizational information, colleague-related questions, general inquiries, and technical support.

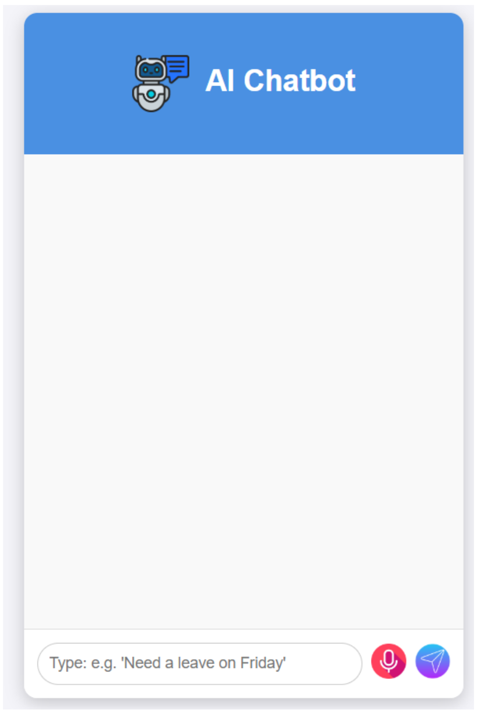
Key functionalities of the chatbot include:

1. Approving leave requests for any specified date or day.
2. Processing shift swap requests between different days.
3. Interpreting RAG inputs and delivering relevant contextual information to employees.
4. Accessing PostgreSQL database tables to provide accurate responses regarding employee schedules.
5. Setting schedule preferences for the upcoming week.

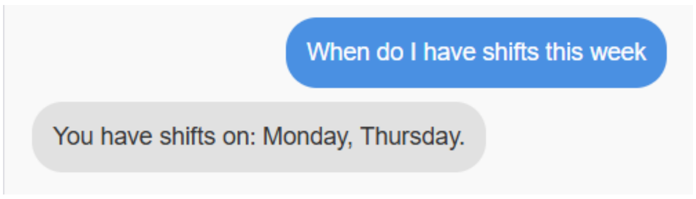
While all these features are also accessible through the graphical user interface (GUI) of the employee portal, the chatbot provides a much simpler and more intuitive alternative. Employees can perform the same actions by merely typing a prompt, allowing the chatbot to handle the underlying processes, thereby reducing manual effort and streamlining the overall experience.

2.3.2. Chatbot UI

The image above displays the chatbot interface integrated within the employee portal. The interface comprises a chatbot window, a text input box, a send button, and a voice command button. In addition to text-based interaction, employees can utilize voice commands, further minimizing the time and effort needed to submit queries or requests, thereby enhancing overall accessibility and user convenience.



The image above displays the chatbot interface integrated within the employee portal. The interface comprises a chatbot window, a text input box, a send button, and a voice command button. In addition to text-based interaction, employees can utilize voice commands, further minimizing the time and effort needed to submit queries or requests, thereby enhancing overall accessibility and user convenience.



When do I have shifts this week

You have shifts on: Monday, Thursday.

2.3.3. Resources used in Chatbot:

1. Flask and supporting libraries:

from flask import Flask, request, jsonify, render_template

These are part of the Flask web framework, used to build lightweight web applications or APIs in Python. We use Flask to create a web server. The request, jsonify, and render_template modules are used to handle incoming requests, return JSON responses, and render HTML templates, respectively.

2. Other supporting libraries:

import os
import re
import psycopg2

We use os to work with environment variables or file paths. We use re for regular expressions, helpful in pattern searching or text processing. We use psycopg2 to connect and interact with PostgreSQL databases using SQL queries directly from Python.

3. Google Generative AI packages:

from google import genai
from google.genai import types

These are part of Google's Generative AI SDK, used to interact with Google's large language models (LLMs), like Gemini 1.5, through Python. We use genai to configure and interact with the model. We use types to send messages in a structured format, to set advanced generation configurations and to work with chat memory or multimodal inputs.

2.3.4. Inputs to the Chatbot

The inputs to the chatbot python code are:

1. RAG document

2. PostgreSQL tables

3. Gemini API key from Google AI Studio

4. Chat prompt template

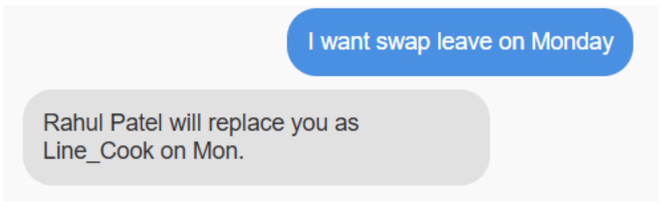
5. Employee name (varies for each employee)

In the employer's portal, the employer uploads a RAG (Retrieval-Augmented Generation) document, which typically contains information that can be disclosed to employees. This includes organizational details such as opening and closing hours, company policies, important announcements, employee roles and responsibilities, technical support FAQs, and answers to common work-related queries.

Retrieval-Augmented Generation (RAG) is a technique that improves the accuracy and relevance of generative AI models by allowing them to retrieve and incorporate information from external sources. In this chatbot system, the uploaded document is processed using RAG to generate accurate, context-aware responses to employee queries based on the available information.

2.3.5. Core Functionalities

Additionally, when an employee submits a leave request for specific day(s), the Python backend accesses the PostgreSQL tables, namely "changed_schedule" and "employees" to determine the employee's scheduled shifts and existing leaves. The chatbot then processes the leave request by optimizing the schedule based on skill ratings through a dedicated leave request algorithm, ensuring efficient management of workforce availability.



I want swap leave on Monday

Rahul Patel will replace you as Line_Cook on Mon.

i. Leave request algorithm:

Input: employee_name, requested_leave_day

1. Verify if employee has a shift (1) on the requested_leave_day:

  If not:

    Return "No shift on this day."

2. Identify employee role and ID from "employees" table.

3. Find replacement:

   - Filter employees with same role.
   - Check availability ('0') on the requested_leave_day.
   - Exclude requesting employee.
   - Sort by skill rating (same role) descending.
   - Select top employee.

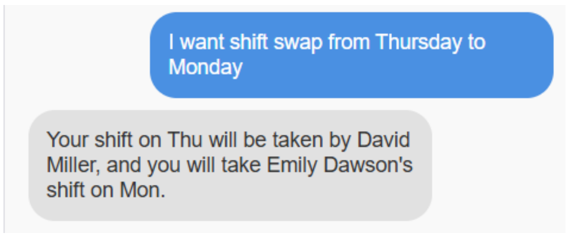
4. Update schedule in "changed_schedule" table:

   - Set requesting employee's shift to '0' (no shift).
   - Set replacement employee's shift to '1' (assigned shift).

5. Return confirmation:

  "[Replacement Employee] will replace you as [Role] on [Day]."

Similarly, when an employee requests a shift swap from one day to another, the Python backend accesses the PostgreSQL tables, “changed_schedule” and “employees” to identify the employee’s current shifts and leave days. It then evaluates the feasibility of the requested swap and processes it by optimizing the schedule based on employee skill ratings. This ensures that the shift swap is handled efficiently while maintaining workforce balance and operational effectiveness, using the same leave request algorithm.


I want shift swap from Thursday to Monday

Your shift on Thu will be taken by David Miller, and you will take Emily Dawson's shift on Mon.

ii. Swap request algorithm:

Input: employee_name, from_day, to_day

1. Validate the swap request:

- Check that employee has shift (1) on from_day.
- Check that employee has no shift (0) on to_day.

If either check fails:

- Return appropriate error message.

2. Find replacement for from_day:

- Same logic as leave request above.

3. Find person to replace on to_day:

- Find employees who have a shift (1) on to_day, same role.
- Exclude requesting employee.
- Sort by skill ascending (lowest skilled).
- Select least skilled employee.

4. Update schedule in “changed_schedule” table:

- Requesting employee:
  from_day → 0, to_day → 1
- Best replacement:
  from_day → 1
- Least skilled employee:
  to_day → 0

5. Return confirmation:

- "Your shift on [from_day] will be taken by [Best Replacement], and you will take [Least Skilled]'s shift on [to_day]."

In general, when a shift swap or replacement request is initiated, the employees who are potential candidates for replacement receive a notification asking for their availability to accept the swap or replacement. The request is approved only if a suitable employee consents to the change. If no employee agrees to the replacement, the request is automatically declined, and the chatbot responds with a message stating, “No suitable employee was found to replace you.”

iii. Schedule Optimization algorithm:

1. Connect to the PostgreSQL database.

2. Read employee limits:

   - From the limits table, retrieve:
     - employee_id, employee_name, min_hours, max_hours
   - Store in a dictionary:
     - Key: employee_id
     - Value: { name, min_hours, max_hours }

3. Read employee preferences:

   - From the preferences table, retrieve:
     - employee_id, employee_name, preferences for mon to sun (0/1)
   - Store as a list of preferred days.

4. For each employee:

   - Calculate total preferred hours = (number of preferred days) × 10
   - If total preferred hours < min_hours:
   - Leave as is (do not add unpreferred days)
   - If total preferred hours > max_hours:
   - Remove preferred days one by one from the end until total hours ≤ max_hours
   - Create final day-wise availability dictionary with values 0 or 1

5. Write optimized schedule:

   - Create new_schedule table if not exists
   - Truncate the table to clear previous data
   - Insert optimized values into new_schedule for each employee

6. Sync to changed_schedule table:

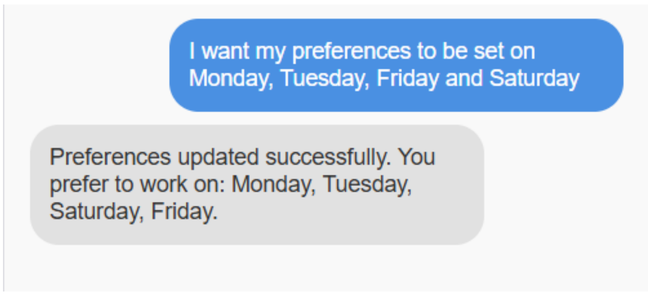
   - Create changed_schedule table if not exists
   - Truncate the table
   - Copy all records from new_schedule into changed_schedule

7. Commit changes and close database connection.

8. Print confirmation:

- "Optimization complete. 'new_schedule' table updated."

Employees can also set their preferred workdays for the upcoming week directly through the chatbot. By simply informing the chatbot of their desired shift days, the system updates the “preferences” table in the PostgreSQL database in real time based on the input received.

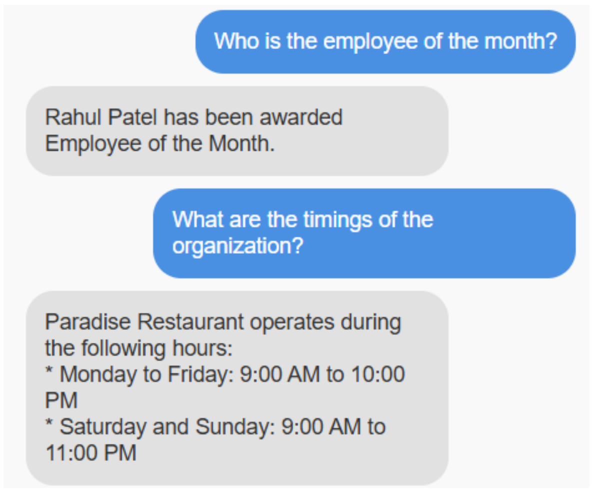


I want my preferences to be set on Monday, Tuesday, Friday and Saturday

Preferences updated successfully. You prefer to work on: Monday, Tuesday, Saturday, Friday.

Once preferences are updated, the Python backend runs an optimization algorithm that considers various constraints to generate an efficient and fair schedule. The optimized results are then stored in the “new_schedule” table, which serves as a blueprint and is updated only once per week. At the start of each week, the “changed_schedule” table—used for day-to-day operations—is initialized with data from the “new_schedule” table. Additionally, any subsequent updates, such as approved leave or swap requests, dynamically modify the “changed_schedule” table by replacing the employee with the most suitable match.

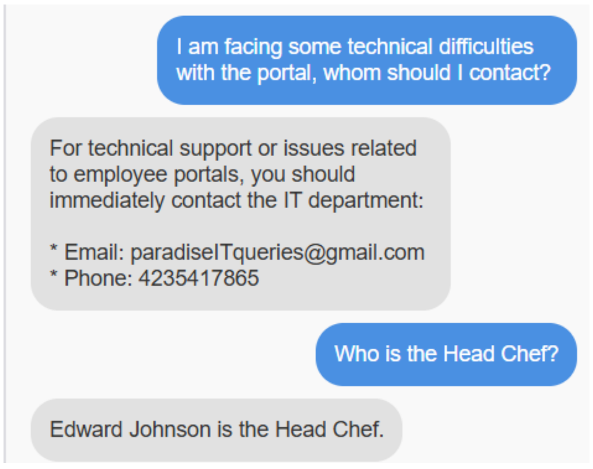
This dynamic scheduling system eliminates the need for manual intervention by managers in routine administrative tasks such as approving leaves, managing shift swaps, setting preferences, or optimizing schedules. As a result, managers can focus on higher-level responsibilities such as supervision and periodically updating employee skill ratings, thereby improving overall operational efficiency.


Who is the employee of the month?

Rahul Patel has been awarded Employee of the Month.

What are the timings of the organization?

Paradise Restaurant operates during the following hours:
* Monday to Friday: 9:00 AM to 10:00 PM
* Saturday and Sunday: 9:00 AM to 11:00 PM


I am facing some technical difficulties with the portal, whom should I contact?

For technical support or issues related to employee portals, you should immediately contact the IT department:

* Email: paradiseITqueries@gmail.com
* Phone: 4235417865

Who is the Head Chef?

Edward Johnson is the Head Chef.

2.4. Business Insights

SailorShift is currently optimized for small and medium-sized enterprises (SMEs), offering a stable, user-friendly solution tailored to their dynamic scheduling needs. Future plans include integrating advanced time-sensitive optimization, enhanced UI/UX, stronger security protocols, and complete cloud scalability ensuring readiness for broader organizational adoption.

2.4.1. Product Capabilities and Client Services

Our scheduling platform automates complex scheduling tasks, minimizing manual intervention and enhancing operational efficiency. Real-time notifications keep both employers and employees promptly informed. A key differentiator is the AI-powered Gemini 2.0 Flash chatbot, which facilitates natural-language interactions for managing leave requests, shift swaps, and schedule inquiries making the system more intuitive and reducing administrative overhead.

2.4.2. Market Positioning and Competitive Advantage

Though developed as a capstone project with limited resources, SailorShift's AI integration provides a clear competitive edge. Unlike traditional systems used by SMEs, our real-time chatbot delivers instant, personalized assistance, streamlining operations and improving responsiveness. This innovation allows smaller businesses access to capabilities typically found in enterprise-level platforms but at a significantly lower cost and complexity.

2.4.3. Target Customers

The system is targeted toward SMEs and local businesses seeking agile, affordable scheduling solutions without the burden of large-scale system complexities. These organizations often struggle with limited resources and require systems that are easy to adopt, adapt, and manage. SailorShift meets these needs directly by offering a flexible and cost-effective alternative with high functionality.

2.4.4. Revenue Model

A SaaS-based subscription model with a minimal initial registration fee and a two-month free trial can be introduced. This pricing strategy encourages adoption and user feedback while enabling sustainable growth through subscription renewals and user retention.

2.4.5. Competitive Landscape

The SME-focused scheduling market is increasingly competitive, with notable players such as Connecteam, Legion, and Soon. Connecteam offers an all-in-one solution with GPS tracking and payroll integration, prioritizing simplicity and cost-effectiveness. Legion leverages intelligent automation to align employee preferences with business needs, enhancing scheduling accuracy and workforce satisfaction. Soon focuses on modern, AI-driven shift planning and time-off management with a sleek user interface.

However, these platforms primarily emphasize backend optimization, often lacking in real-time, conversational AI capabilities. SailorShift's key differentiator is its Gemini 2.0 Flash-powered chatbot, which enables immediate, interactive assistance for scheduling tasks. This feature enhances usability and streamlines communication between employees and managers, offering a personalized experience typically absent in traditional solutions. Our chatbot-centric model delivers intelligent support on-demand, positioning SailorShift as a uniquely responsive and efficient tool in the SME scheduling space

3. EVALUATION

3.1. Design vs. Implementation

At the outset, our objective was to design a scalable, fully automated scheduling system capable of handling complex tasks like leave management, shift swaps, and mass onboarding, with the potential for large-scale enterprise integration. While we successfully developed two functional web portals for employers and employees, along with a robust AI chatbot powered by Gemini 2.0 Flash, the final product diverged from our original blueprint. The envisioned system for enterprise-level deployment faced practical constraints, particularly integration with legacy systems and handling high concurrent requests. As development progressed, we pivoted toward a solution optimized for SMEs and local businesses, delivering streamlined functionalities that better suited our resource and time limitations.

Key differences emerged due to limited access to real-world enterprise data, tighter project timelines, and stakeholder feedback from SMEs. User testing revealed unexpected behavior users relied heavily on the chatbot not just for basic queries but for complex tasks like schedule modifications and leave approvals. This contradicted our initial assumption that the chatbot would only supplement the GUI. Consequently, we were compelled to extend the chatbot's backend integration mid-development. We also discovered that real-time synchronization between employee and employer platforms required more robust architecture than originally planned, pushing us to invest time in reinforcing data consistency and update reliability.

These challenges underscore the principle illustrated by the "Cost of Change" curve. The cost of implementing changes rises exponentially as development progresses from requirements gathering to production. In our case, late-stage enhancements to chatbot functionality and architecture improvements significantly increased development overhead. If these needs had been anticipated during the design phase, the effort and cost involved would have been considerably lower. This reinforces the importance of early requirement validation, modular design planning, and iterative feedback loops.

Ultimately, while the project met core goals for SMEs, it highlighted the value of agile thinking, early-stage planning, and proactive adaptation. These insights not only enhanced the quality of our deliverable but also prepared us for handling real-world complexities in future system design and implementation roles.

3.2. Entrepreneurship Opportunities

SailorShift presents a promising entrepreneurial opportunity in the growing market of workforce management, especially for small and medium-sized enterprises (SMEs) and local businesses. These organizations often lack access to affordable, intelligent scheduling systems and represent our ideal initial customer base. Beyond SMEs, industries such as healthcare, education, retail, hospitality, and small-scale manufacturing, which require dynamic scheduling, also show strong potential for adoption. Our AI-powered scheduling system, with its real-time chatbot support, simplifies workforce management without high infrastructure or training costs, making it highly appealing to these customer segments.

From an investment perspective, our project is well-suited for early-stage funding from angel investors, seed-stage venture capitalists, and technology incubators. The uniqueness of integrating conversational AI for scheduling automation provides a distinct market differentiation. We also anticipate support from local startup accelerators, government grants aimed at boosting SME digitization, and university innovation funds. Additionally, crowdfunding platforms like Kickstarter could help validate product-market fit while raising initial capital. These funding options would help us scale the platform, refine features, and expand into untapped customer segments.

Regarding intellectual property, our core advantage lies in the proprietary algorithms and the integration of AI-driven decision-making processes within real-time scheduling. While patent protection may be less practical due to software's fast-paced evolution, copyright or trade secret protection could safeguard our backend logic and chatbot workflows. Competitively, platforms like Connecteam, Legion, and Soon offer similar scheduling services, but their lack of real-time conversational AI support distinguishes our offering. Currently in the introduction phase of the product life cycle, SailorShift’s next steps focus on increasing market visibility, customer onboarding, and product refinement. As adoption grows, continuous innovation and strategic partnerships will drive us into the growth and maturity phases, maintaining a competitive edge in a rapidly evolving scheduling tech market.

CONCLUSION

Our capstone project achieved its core goal of designing and deploying an automated workforce scheduling system tailored to the needs of small and medium-sized enterprises (SMEs). We successfully developed two integrated web applications—one for employers and another for employees capable of automating leave requests, shift swaps, and weekly scheduling based on user preferences. The integration of the Gemini 2.0 Flash-powered AI chatbot was a key success, offering real-time conversational support and significantly improving user engagement. By reducing administrative overhead, improving transparency, and enabling efficient workforce management, SailorShift was validated through testing with local businesses and demonstrated clear practical value in SME environments.

Despite these successes, the project encountered challenges in achieving enterprise-level scalability. Our initial design aimed to support large-scale deployments, but limitations in integrating with diverse legacy systems and managing high volumes of concurrent users proved more complex than expected. If granted more time and resources, we would explore scalable cloud infrastructure such as AWS or Azure, strengthen database capacity, and implement modular microservices architecture to ensure robust, high-performance deployment at scale. Additionally, while the chatbot was initially designed for simple queries, user testing revealed demand for more advanced conversational capabilities. This insight would guide further development to improve AI reasoning, deeper database interaction, and multi-turn dialog flow for handling complex scheduling scenarios.

Looking ahead, several enhancements are planned to elevate SailorShift's capabilities and competitiveness. Future iterations will include mobile app integration, payroll processing, compliance monitoring, and advanced analytics dashboards for deeper workforce insights. To boost adoption and market reach, partnerships with tech providers and industry stakeholders will be explored. Moreover, continuous user training and support will be critical to ensure users fully benefit from the system's capabilities. With these enhancements, SailorShift can transition from a student capstone to a market-ready product with long-term scalability, solidifying its position in the evolving SME scheduling solutions landscape.

REFERENCES

1. Designing a text-based AI scheduling assistant chatbot for a business environment. https://www.diva-portal.org/smash/get/diva2%3A1647530/FULLTEXT01.pdf?utm_source=chatgpt.com

2. The general employee scheduling problem. An integration of MS and AI. https://www.researchgate.net/publication/220472067_The_general_employee_scheduling_problem_An_integration_of_MS_and_AI

3. Connectteam website - https://connecteam.com/

4. Soon website - https://www.soon.works/

5. Legion website - https://legion.co/

6. Calendar.help: Designing a Workflow-Based Scheduling Agent with Humans in the Loop - https://arxiv.org/abs/1703.08428?utm_source=chatgpt.com

7. Integrated employee scheduling with known employee demand, including breaks, overtime, and employee preferences - https://www.researchgate.net/publication/345032131_Integrated_employee_scheduling_with_known_employee_demand_including_breaks_overtime_and_employee_preferences

8. Rippling website - https://www.rippling.com/en-CA

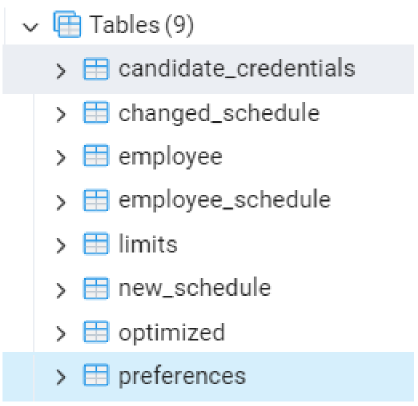
9. A fast-flexible strategy based approach to solving employee scheduling problem considering soft work time - https://www.nature.com/articles/s41598-024-56745-4?utm_source=chatgpt.com

10. Sample Schedule data collected from Restaurant il focolaio, du Square-Phillips, Montréal.

11. https://static.projectmanagement.com/images/CostChange11.jpg

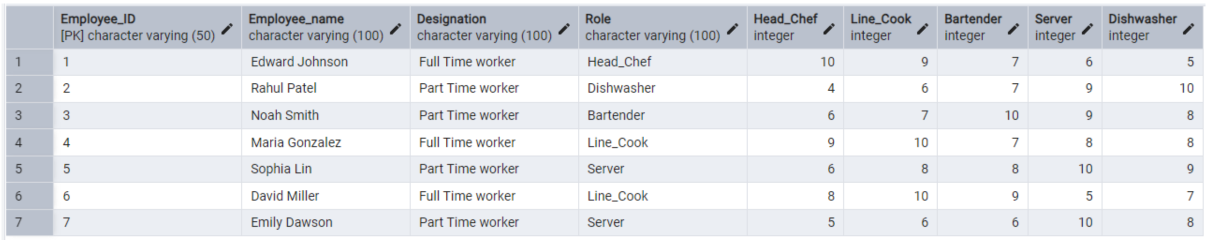
APPENDIX

I. PostgreSQL Table Schema:



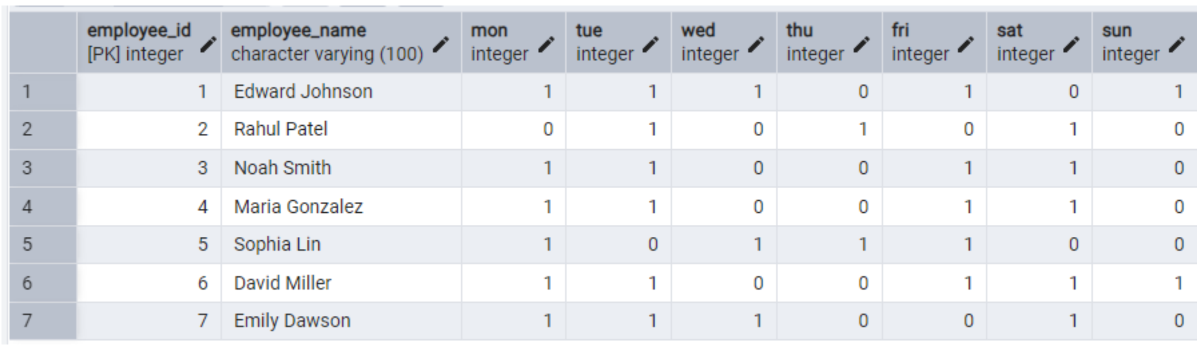
Tables (9)
- candidate_credentials
- changed_schedule
- employee
- employee_schedule
- limits
- new_schedule
- optimized
- preferences

i. employee table:



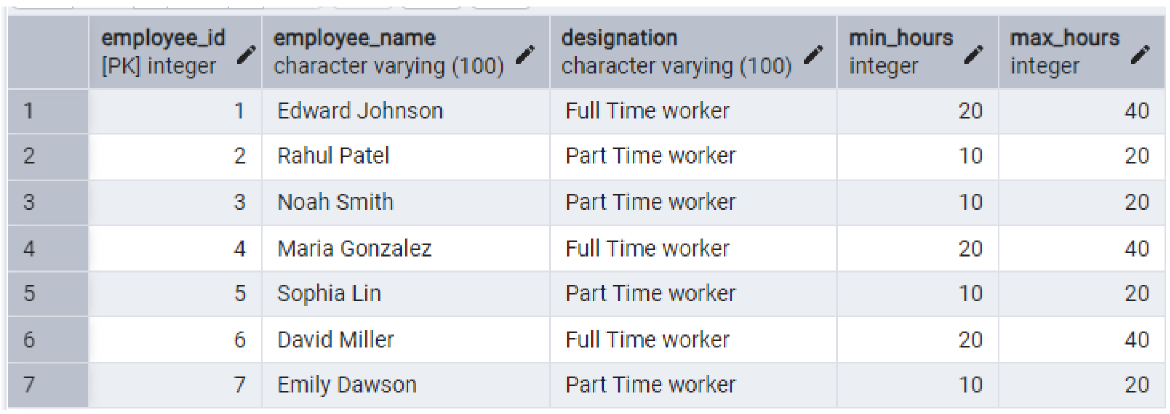
	Employee_ID [PK] character varying (50)	Employee_name character varying (100)	Designation character varying (100)	Role character varying (100)	Head_Chef integer	Line_Cook integer	Bartender integer	Server integer	Dishwasher integer
1	1	Edward Johnson	Full Time worker	Head_Chef	10	9	7	6	5
2	2	Rahul Patel	Part Time worker	Dishwasher	4	6	7	9	10
3	3	Noah Smith	Part Time worker	Bartender	6	7	10	9	8
4	4	Maria Gonzalez	Full Time worker	Line_Cook	9	10	7	8	8
5	5	Sophia Lin	Part Time worker	Server	6	8	8	10	9
6	6	David Miller	Full Time worker	Line_Cook	8	10	9	5	7
7	7	Emily Dawson	Part Time worker	Server	5	6	6	10	8

ii. preferences table:



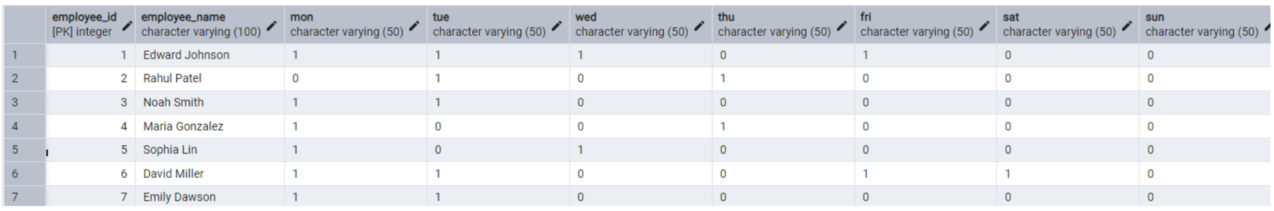
	employee_id [PK] integer	employee_name character varying (100)	mon integer	tue integer	wed integer	thu integer	fri integer	sat integer	sun integer
1	1	Edward Johnson	1	1	1	0	1	0	1
2	2	Rahul Patel	0	1	0	1	0	1	0
3	3	Noah Smith	1	1	0	0	1	1	0
4	4	Maria Gonzalez	1	1	0	0	1	1	0
5	5	Sophia Lin	1	0	1	1	1	0	0
6	6	David Miller	1	1	0	0	1	1	1
7	7	Emily Dawson	1	1	1	0	0	1	0

iii. limits table:



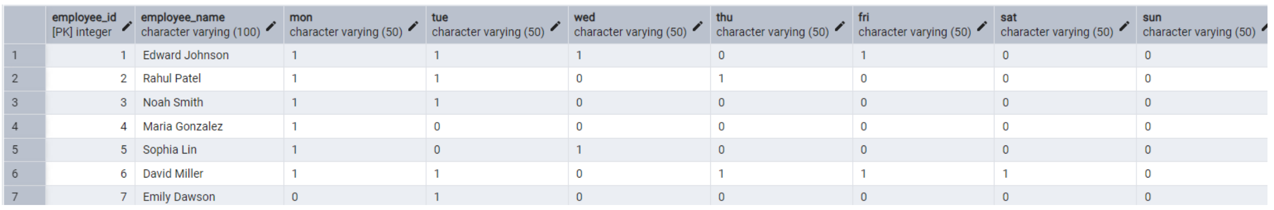
	employee_id [PK] integer	employee_name character varying (100)	designation character varying (100)	min_hours integer	max_hours integer
1	1	Edward Johnson	Full Time worker	20	40
2	2	Rahul Patel	Part Time worker	10	20
3	3	Noah Smith	Part Time worker	10	20
4	4	Maria Gonzalez	Full Time worker	20	40
5	5	Sophia Lin	Part Time worker	10	20
6	6	David Miller	Full Time worker	20	40
7	7	Emily Dawson	Part Time worker	10	20

iv. new_schedule:



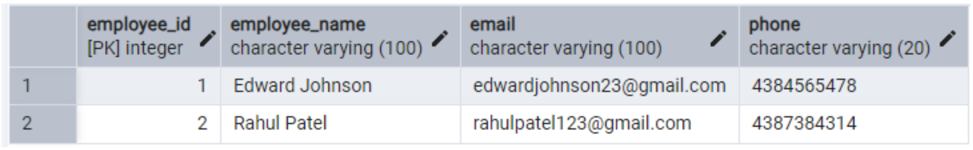
	employee_id [PK] integer	employee_name character varying (100)	mon character varying (50)	tue character varying (50)	wed character varying (50)	thu character varying (50)	fri character varying (50)	sat character varying (50)	sun character varying (50)
1	1	Edward Johnson	1	1	1	0	1	0	0
2	2	Rahul Patel	0	1	0	1	0	0	0
3	3	Noah Smith	1	1	0	0	0	0	0
4	4	Maria Gonzalez	1	0	0	1	0	0	0
5	5	Sophia Lin	1	0	1	0	0	0	0
6	6	David Miller	1	1	0	0	1	1	0
7	7	Emily Dawson	1	1	0	0	0	0	0

v. changed_schedule:



	employee_id [PK] integer	employee_name character varying (100)	mon character varying (50)	tue character varying (50)	wed character varying (50)	thu character varying (50)	fri character varying (50)	sat character varying (50)	sun character varying (50)
1	1	Edward Johnson	1	1	1	0	1	0	0
2	2	Rahul Patel	1	1	0	1	0	0	0
3	3	Noah Smith	1	1	0	0	0	0	0
4	4	Maria Gonzalez	1	0	0	0	0	0	0
5	5	Sophia Lin	1	0	1	0	0	0	0
6	6	David Miller	1	1	0	1	1	1	0
7	7	Emily Dawson	0	1	0	0	0	0	0

vi. candidate_credentials:



	employee_id [PK] integer	employee_name character varying (100)	email character varying (100)	phone character varying (20)
1	1	Edward Johnson	edwardjohnson23@gmail.com	4384565478
2	2	Rahul Patel	rahulpatel123@gmail.com	4387384314

II. Python code for Optimization:

```
import psycopg2
def get_db_connection():
    """Helper to connect to Postgres."""
    return psycopg2.connect(
        dbname="postgres",
        user="postgres",
        password="****",
        host="****",
        port="5432"
    )
def optimize_schedule():
    """
    1. Read from limits table: employee_id, employee_name, min_hours, max_hours
    2. Read from preferences table: mon..sun (0/1)
    3. For each employee, keep or remove days to respect min/max hours.
       One day = 10 hours.
    4. Write final 0/1 schedule to new_schedule.
    """
    DAYS = ["mon","tue","wed","thu","fri","sat","sun"]
    HOURS_PER_DAY = 10
```

```
conn = get_db_connection()
cur = conn.cursor()
# 1) Read from limits
cur.execute("""
    SELECT employee_id, employee_name, designation, min_hours, max_hours
      FROM limits
     ORDER BY employee_id
""")
limit_rows = cur.fetchall()
# store in a dict:  emp_id -> { "name":..., "min":..., "max":... }
emp_limits = {}
for row in limit_rows:
    eid, ename, designation, min_h, max_h = row
    emp_limits[eid] = {
        "name": ename,
        "min": min_h,
        "max": max_h
    }
# 2) Read from preferences table
#    Each row has: employee_id, employee_name, mon..sun (ints 0/1)
cur.execute("""
    SELECT employee_id, employee_name, mon, tue, wed, thu, fri, sat, sun
      FROM preferences
     ORDER BY employee_id
""")
pref_rows = cur.fetchall()

# We'll store final optimized schedule in a dictionary,
# keyed by employee_id => {"name":..., "days": {"mon":0/1, ...}}
final_schedule = {}

for row in pref_rows:
    eid = row[0]
    ename = row[1]
    # days_avail is e.g. [1,1,0,1,1,0,1]
    days_avail = list(row[2:])  # mon..sun
    day_map = dict(zip(DAYS, days_avail))
```

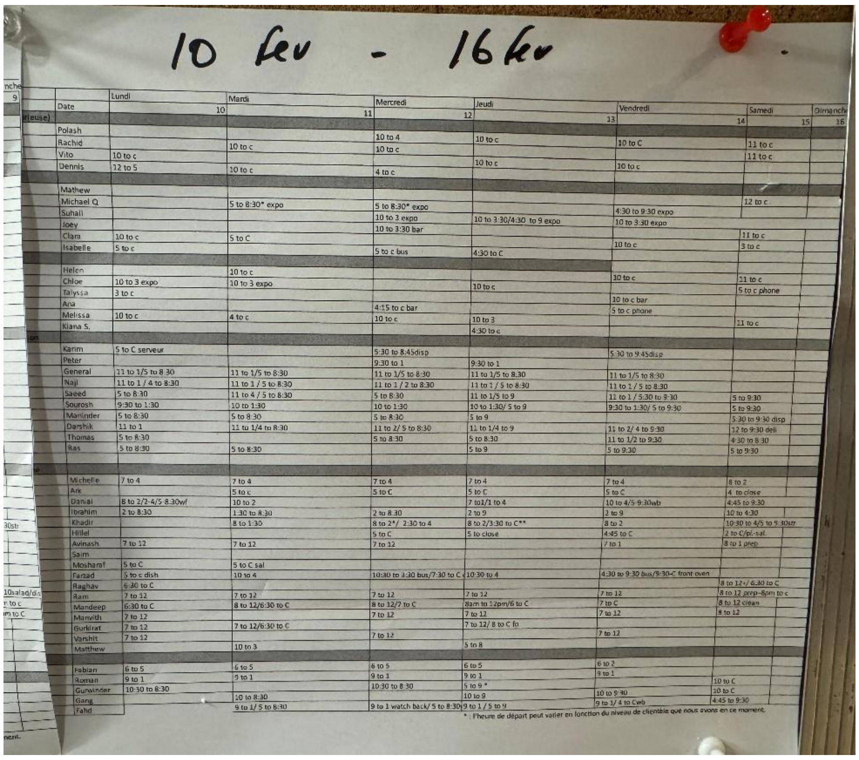
```
# If employee is in limits table, get min/max
if eid not in emp_limits:
    # If somehow not found, skip
    continue
min_h = emp_limits[eid]["min"]
max_h = emp_limits[eid]["max"]
# convert the 0/1 availability into a *tentative* schedule
#  if day_map[day]==1 => we plan to schedule them that day
# each scheduled day is 10 hours
scheduled_days = [d for d in DAYS if day_map[d] == 1]
total_pref_hours = len(scheduled_days) * HOURS_PER_DAY
# A) If total preferred < min, you *could* add days not in preference
#    But let's do a simple approach: we won't forcibly add days
#    (or you can decide to add them if you truly need to meet min).
if total_pref_hours < min_h:
    # Not meeting min. We could add days if you want:
    #   for day in DAYS:
    #       if day_map[day]==0 => consider adding
    # But in many real setups, if they didn't prefer it, we don't assign it.
    pass

# B) If total preferred > max, remove days until within max
#    We'll remove from "least necessary" day. For example, remove from
#    the day that has the highest overall coverage or from random day.
#    For simplicity, remove from the end until we meet max:
while total_pref_hours > max_h and scheduled_days:
    # pick a day to remove. E.g. remove the last one in the list
    # or you can remove the day with highest coverage, etc.
    day_to_remove = scheduled_days[-1]  # remove last
    scheduled_days.pop()
    total_pref_hours = len(scheduled_days) * HOURS_PER_DAY
# Now we have a final set of scheduled days for this employee
# build a dict of 0/1 for each day
final_days = {d: 1 if d in scheduled_days else 0 for d in DAYS}
# store in final_schedule
final_schedule[eid] = {
    "name": ename,
            "days": final_days
        }
    # 3) Write final schedule to new_schedule table
    #    First create new_schedule if not exists
    cur.execute("""
        CREATE TABLE IF NOT EXISTS new_schedule (
            employee_id INT PRIMARY KEY,
            employee_name VARCHAR(100),
            mon VARCHAR(50),
            tue VARCHAR(50),
            wed VARCHAR(50),
            thu VARCHAR(50),
            fri VARCHAR(50),
            sat VARCHAR(50),
            sun VARCHAR(50)
        );
    """)
    # optional: clear out the table each time, or do an upsert
    cur.execute("TRUNCATE TABLE new_schedule;")

    for eid, data in final_schedule.items():
        ename = data["name"]
        day_vals = data["days"]  # {mon:0/1, tue:0/1, ...}
        # insert as strings "0" or "1"
        cur.execute("""
            INSERT INTO new_schedule (employee_id, employee_name, mon, tue, wed, thu, fri, sat,
sun)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
        """, (
            eid,
            ename,
            str(day_vals["mon"]),
            str(day_vals["tue"]),
            str(day_vals["wed"]),
            str(day_vals["thu"]),
            str(day_vals["fri"]),
            str(day_vals["sat"]),
            str(day_vals["sun"])
        ))
            # --- Sync to changed_schedule ---
        cur.execute("""
            CREATE TABLE IF NOT EXISTS changed_schedule (
                employee_id INT PRIMARY KEY,
                employee_name VARCHAR(100),
                mon VARCHAR(50),
                tue VARCHAR(50),
                wed VARCHAR(50),
                thu VARCHAR(50),
                fri VARCHAR(50),
                sat VARCHAR(50),
                sun VARCHAR(50)
            );
        """)
        cur.execute("TRUNCATE TABLE changed_schedule;")
        cur.execute("""
            INSERT INTO changed_schedule (employee_id, employee_name, mon, tue, wed, thu, fri,
sat, sun)
            SELECT employee_id, employee_name, mon, tue, wed, thu, fri, sat, sun
            FROM new_schedule
        """)
    conn.commit()
    cur.close()
    conn.close()
    print("Optimization complete. 'new_schedule' table updated.")
if __name__ == "__main__":
    optimize_schedule()
```

III. Sample schedule data collected from a Restaurant

The following data has been collected from Restaurant il focolaio, du Square-Phillips, Montréal.



10 fev - 16 fev

		Lundi	Mardi	Mercredi	Jeudi	Vendredi	Samedi	Dimanche
	Date	10	11	12	13	14	15	16
(euse)								
	Polash			10 to 4	10 to c	10 to C	11 to c	
	Rachid		10 to c	10 to c			11 to c	
	Vito	10 to c			10 to c	10 to c		
	Dennis	12 to 5	10 to c	4 to c				
	Mathew						12 to c	
	Michael Q		5 to 8:30* expo	5 to 8:30* expo		4:30 to 9:30 expo		
	Suhail			10 to 3 expo	10 to 3:30/4:30 to 9 expo	10 to 3:30 expo		
	Joey			10 to 3:30 bar			11 to c	
	Clara	10 to c	5 to C			10 to c	3 to c	
	Isabelle	5 to c		5 to c bus	4:30 to C			
	Helen		10 to c			10 to c	11 to c	
	Chloe	10 to 3 expo	10 to 3 expo		10 to c		5 to c phone	
	Talyssa	3 to c				10 to c bar		
	Ana			4:15 to c bar		5 to c phone		
	Melissa	10 to c	4 to c	10 to c	10 to 3		11 to c	
	Kiana S.				4:30 to c			
ion								
	Karim	5 to C serveur		5:30 to 8:45disp		5:30 to 9:45disp		
	Peter			9:30 to 1	9:30 to 1			
	General	11 to 1/5 to 8:30	11 to 1/5 to 8:30	11 to 1/5 to 8:30	11 to 1/5 to 8:30	11 to 1/5 to 8:30		
	Naji	11 to 1 / 4 to 8:30	11 to 1 / 5 to 8:30	11 to 1 / 2 to 8:30	11 to 1 / 5 to 8:30	11 to 1 / 5 to 8:30		
	Saeed	5 to 8:30	11 to 4 / 5 to 8:30	5 to 8:30	11 to 1/5 to 9	11 to 1 / 5:30 to 9:30	5 to 9:30	
	Sourosh	9:30 to 1:30	10 to 1:30	10 to 1:30	10 to 1:30/ 5 to 9	9:30 to 1:30/ 5 to 9:30	5 to 9:30	
	Maninder	5 to 8:30	5 to 8:30	5 to 8:30	5 to 9		5:30 to 9:30 disp	
	Darshik	11 to 1	11 to 1/4 to 8:30	11 to 2/ 5 to 8:30	11 to 1/4 to 9	11 to 2/ 4 to 9:30	12 to 9:30 deli	
	Thomas	5 to 8:30		5 to 8:30	5 to 8:30	11 to 1/2 to 9:30	4:30 to 8:30	
	Ras	5 to 8:30	5 to 8:30		5 to 9	5 to 9:30	5 to 9:30	
	Michelle	7 to 4	7 to 4	7 to 4	7 to 4	7 to 4	6 to 2	
	Ark		5 to c	5 to C	5 to C	5 to C	4 to close	
	Danial	8 to 2/2-4/5-8:30wf	10 to 2		7 to1/1 to 4	10 to 4/5-9:30wb	4:45 to 9:30	
	Ibrahim	2 to 8:30	1:30 to 8:30	2 to 8:30	2 to 9	2 to 9	10 to 4:30	
	Khadir		8 to 1:30	8 to 2*/ 2:30 to 4	8 to 2/3:30 to C**	8 to 2	10:30 to 4/5 to 9:30str	
	Hillel			5 to C	5 to close	4:45 to C	2 to C/pl.-sal.	
	Avinash	7 to 12	7 to 12	7 to 12		7 to 1	8 to 1 prep	
	Saim							
	Mosharaf	5 to C	5 to C sal					
	Farzad	5 to c dish	10 to 4	10:30 to 3:30 bus/7:30 to C	10:30 to 4	4:30 to 9:30 bus/9:30-C front oven		
	Raghav	6:30 to C					8 to 12+/ 6:30 to C	
	Ram	7 to 12	7 to 12	7 to 12	7 to 12	7 to 12	8 to 12 prep-8pm to c	
	Mandeep	6:30 to C	8 to 12/6:30 to C	8 to 12/7 to C	8am to 12pm/6 to C	7 to C	8 to 12 clean	
	Manvith	7 to 12		7 to 12	7 to 12	7 to 12	8 to 12	
	Gurkirat	7 to 12	7 to 12/6:30 to C		7 to 12/ 8 to C fo			
	Varshit	7 to 12		7 to 12		7 to 12		
	Matthew		10 to 3		5 to 8			
	Fabian	6 to 5	6 to 5	6 to 5	6 to 5	6 to 2		
	Roman	9 to 1	9 to 1	9 to 1	9 to 1	9 to 1		
	Gurwinder	10:30 to 8:30		10:30 to 8:30	5 to 9 *		10 to C	
	Gang		10 to 8:30		10 to 9	10 to 9:30	10 to C	
	Fahd		9 to 1/ 5 to 8:30	9 to 1 watch back/ 5 to 8:30	9 to 1 / 5 to 9	9 to 1/ 4 to Cwb	4:45 to 9:30	

* : l'heure de départ peut varier en fonction du niveau de clientèle que nous avons en ce moment.

IV. RAG Document Sample:

The following sample document has been used to input the RAG in Chatbot to get contextual responses from the Chatbot.

-----------------RAG Document Starts here-------------

Welcome to Paradise Restaurant!

Paradise Restaurant proudly stands as the premier destination for authentic Indian cuisine located in the bustling heart of downtown Montreal. For over eight years, our establishment has earned a stellar reputation for its outstanding food quality, exceptional customer service, and warm, welcoming atmosphere. As an integral part of our dedicated staff, your contribution is crucial in continuing this tradition of excellence. This document will serve as a comprehensive guide to provide you, our valued employee, with all essential information required to excel in your role.

Restaurant Overview: Paradise Restaurant is centrally located in downtown Montreal, easily accessible by public transportation and surrounded by various attractions, businesses, and shops. This strategic location ensures a high footfall, especially during lunch and dinner hours. Given our prime spot and stellar reputation, it's imperative that we maintain the highest standards in food preparation, presentation, cleanliness, and customer interaction.

Team and Roles: Our team comprises dedicated professionals passionate about providing exceptional dining experiences. Edward Johnson serves as our Head Chef, bringing over 15 years of culinary expertise and leadership to our kitchen. His wealth of knowledge in Indian culinary traditions ensures that each dish served maintains authenticity and excellence. As Head Chef, Edward is responsible for menu creation, food quality supervision, kitchen management, and mentoring kitchen staff.

Rahul Patel has been awarded Employee of the Month for his remarkable dedication and versatility. Rahul has significantly contributed to multiple roles, ensuring seamless operations even during peak hours. His commitment exemplifies the teamwork and flexibility valued at Paradise Restaurant.

Other team members include Noah Smith (Bartender), Maria Gonzalez (Line Cook), Sophia Lin (Server), David Miller (Line Cook), and Emily Dawson (Server). Each employee plays a vital role in maintaining the restaurant's reputation. Cooperation among team members is essential to creating a harmonious and efficient work environment.

Operational Hours: Paradise Restaurant operates during the following hours:

- Monday to Friday: 9:00 AM to 10:00 PM
- Saturday and Sunday: 9:00 AM to 11:00 PM

Employees are required to arrive at least 15 minutes before their scheduled shifts to facilitate a smooth transition between shifts and ensure preparations are complete for the day's service. Punctuality is highly valued as it directly impacts the restaurant's ability to serve customers efficiently.

Employee Roles and Responsibilities: Each role within Paradise Restaurant carries specific responsibilities essential for maintaining our high operational standards.

- Head Chef: Edward Johnson oversees kitchen operations, menu planning, food preparation quality, and compliance with health regulations. He coordinates closely with line cooks to ensure kitchen efficiency.
- Line Cooks: Maria Gonzalez and David Miller are responsible for preparing ingredients, maintaining kitchen cleanliness, assisting in cooking dishes under the chef's guidance, and adhering strictly to food safety standards.
- Bartender: Noah Smith handles beverage preparation, inventory management for drinks, maintaining cleanliness and organization of the bar area, and ensuring compliance with local alcohol serving regulations.
- Servers: Sophia Lin and Emily Dawson greet customers, take orders, provide menu recommendations, ensure timely food and beverage delivery, handle payments, and address customer queries or concerns efficiently and politely.
- Dishwasher/All-rounder: Rahul Patel's versatile role includes dishwashing, maintaining kitchen cleanliness, occasionally assisting servers and cooks during busy periods, and ensuring smooth operational flow.

Skill Ratings and Performance Management: Employees are regularly evaluated based on skill levels, teamwork, punctuality, customer interactions, and contributions to overall operations. Regular assessments help identify opportunities for training and improvement. Employees showing exemplary performance, like Rahul Patel, are recognized through initiatives such as "Employee of the Month," fostering a culture of motivation and continuous improvement.

Health, Safety, and Hygiene: Adherence to stringent health, safety, and hygiene standards is non-negotiable. Regular hand washing, proper uniform maintenance, and cleanliness in personal and shared workspaces are mandatory. Monthly inspections ensure compliance with health codes, and non-adherence can result in disciplinary actions or additional training.

Technical Assistance and Support: For technical support or issues related to employee portals, point-of-sale systems, scheduling software, or any technological difficulties, employees should immediately contact the IT department:

- Email: paradiseITqueries@gmail.com
- Phone: 4235417865

Prompt reporting of technical issues ensures minimal operational disruption.

General Queries and Concerns: For any other non-technical queries, work-related concerns, suggestions, or personal issues affecting your work performance, please contact the restaurant manager directly:

- Email: manager.paradise@gmail.com
- Phone: 4587256841

Employee Communication and Chatbot Integration: Our employee chatbot system integrated into the employee portal is designed to enhance your workplace experience. The chatbot can assist with:

- Checking personal schedules and shifts from the database.
- Confirming shifts for specific dates or days.
- Answering general restaurant information queries and guiding you toward resources or contacts for further assistance.

Ensure all interactions with the chatbot are professional and clear for accurate responses. The chatbot also recognizes casual language and maintains context, ensuring smooth and helpful interactions.

Training and Development: Paradise Restaurant prioritizes continuous learning and development. Regular workshops, training sessions, and briefings are conducted to enhance skill sets, customer interaction skills, and culinary knowledge. Active participation in these programs is encouraged and often recognized in performance evaluations.

Customer Interaction: Employees should consistently practice excellent customer interaction skills. Listen actively, remain polite, handle complaints calmly, and ensure customers leave with a positive impression. Your actions reflect directly on Paradise Restaurant's reputation.

Emergency Procedures: All employees should familiarize themselves with emergency evacuation routes, first aid kit locations, and emergency contacts displayed prominently within the restaurant premises. Regular emergency drills are conducted to ensure preparedness and employee safety.

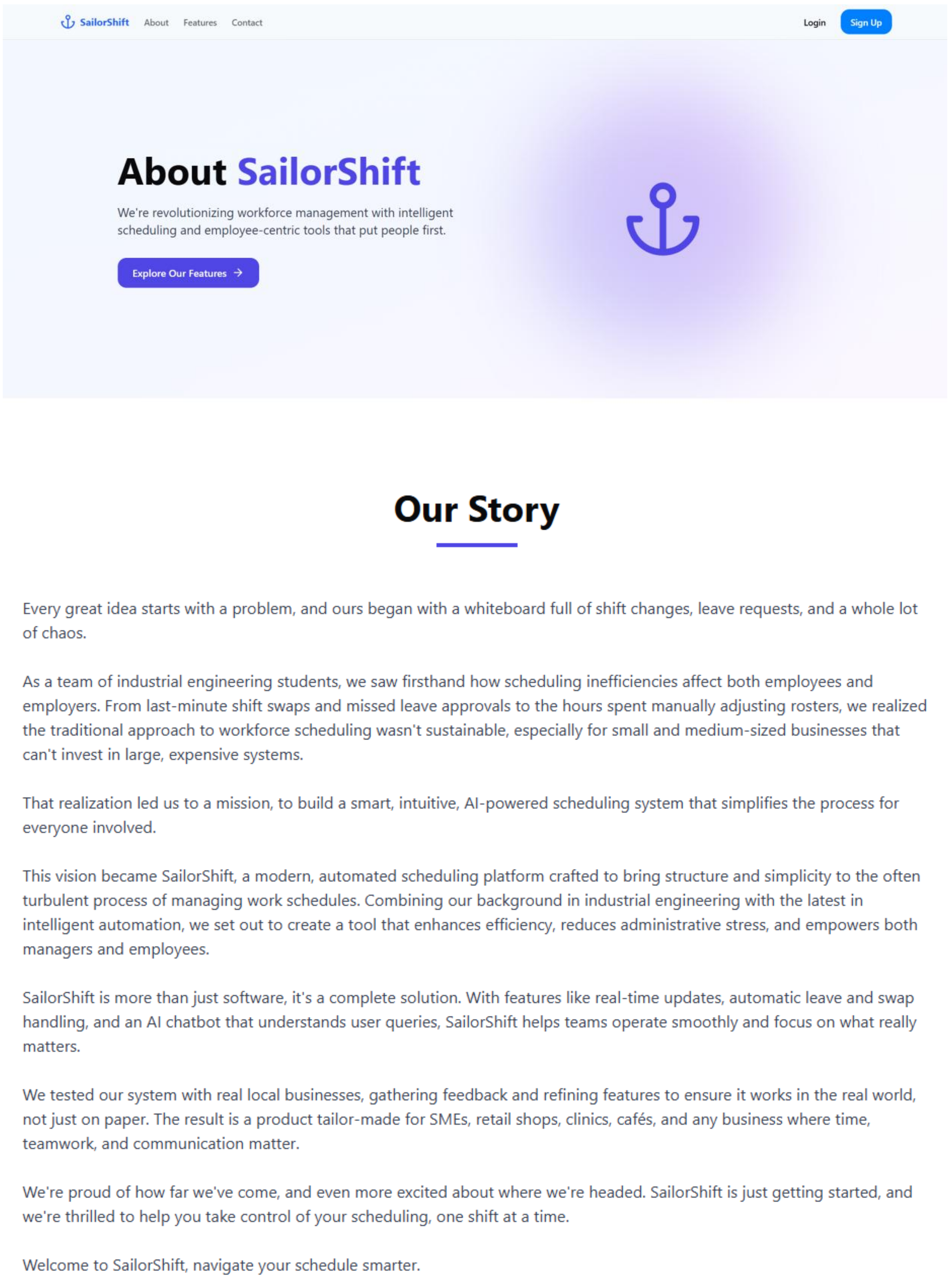
Conclusion: Paradise Restaurant values your role and contributions highly. By following these guidelines, you help maintain the restaurant's tradition of excellence. Remember, effective communication, teamwork, punctuality, adherence to standards, and a positive attitude are fundamental to our continued success. Thank you for being an essential part of Paradise Restaurant’s dedicated team!

-----------------RAG Document Ends here-------------

V. ShiftSync website URL

https://sailor-shift.lovable.app/

VI. ShiftSync website pages



About SailorShift

We're revolutionizing workforce management with intelligent scheduling and employee-centric tools that put people first.

Our Story

Every great idea starts with a problem, and ours began with a whiteboard full of shift changes, leave requests, and a whole lot of chaos.

As a team of industrial engineering students, we saw firsthand how scheduling inefficiencies affect both employees and employers. From last-minute shift swaps and missed leave approvals to the hours spent manually adjusting rosters, we realized the traditional approach to workforce scheduling wasn't sustainable, especially for small and medium-sized businesses that can't invest in large, expensive systems.

That realization led us to a mission, to build a smart, intuitive, AI-powered scheduling system that simplifies the process for everyone involved.

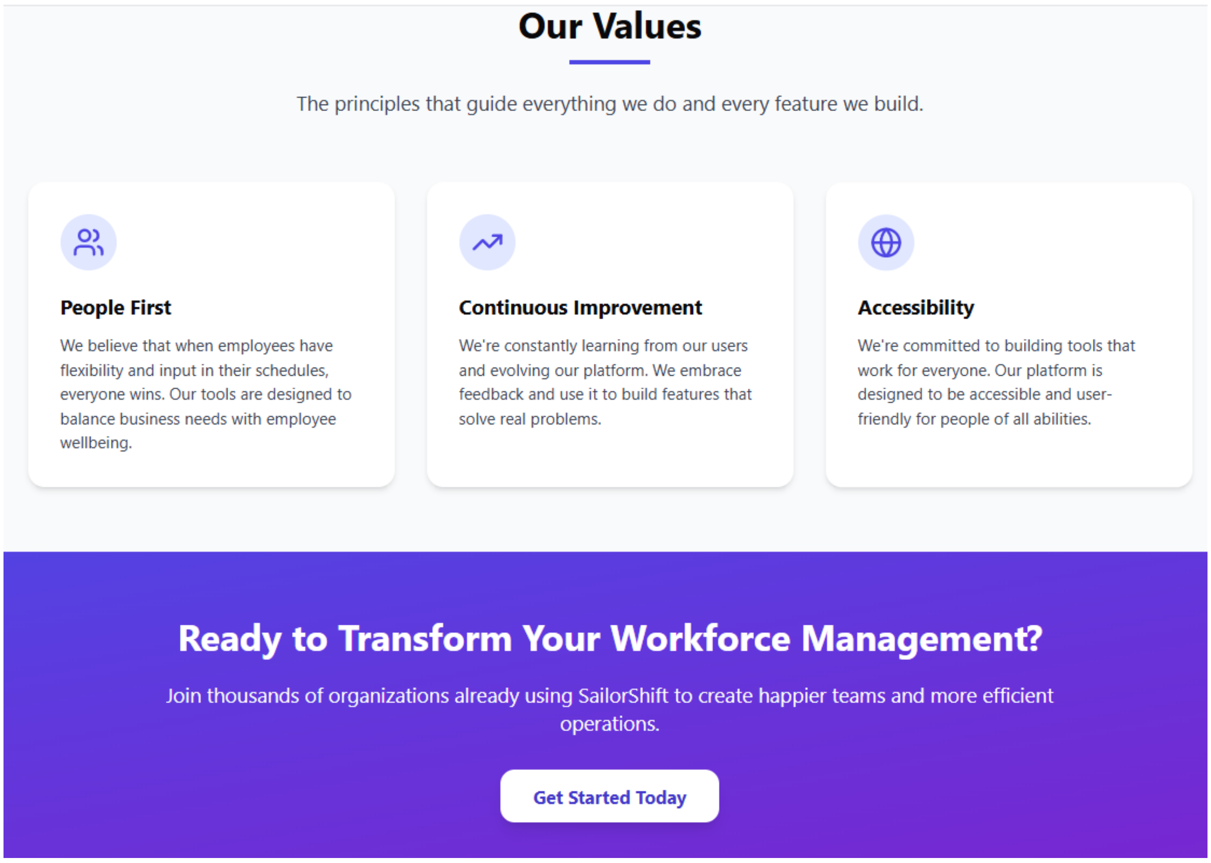
This vision became SailorShift, a modern, automated scheduling platform crafted to bring structure and simplicity to the often turbulent process of managing work schedules. Combining our background in industrial engineering with the latest in intelligent automation, we set out to create a tool that enhances efficiency, reduces administrative stress, and empowers both managers and employees.

SailorShift is more than just software, it's a complete solution. With features like real-time updates, automatic leave and swap handling, and an AI chatbot that understands user queries, SailorShift helps teams operate smoothly and focus on what really matters.

We tested our system with real local businesses, gathering feedback and refining features to ensure it works in the real world, not just on paper. The result is a product tailor-made for SMEs, retail shops, clinics, cafés, and any business where time, teamwork, and communication matter.

We're proud of how far we've come, and even more excited about where we're headed. SailorShift is just getting started, and we're thrilled to help you take control of your scheduling, one shift at a time.

Welcome to SailorShift, navigate your schedule smarter.



Our Values

The principles that guide everything we do and every feature we build.

People First

We believe that when employees have flexibility and input in their schedules, everyone wins. Our tools are designed to balance business needs with employee wellbeing.

Continuous Improvement

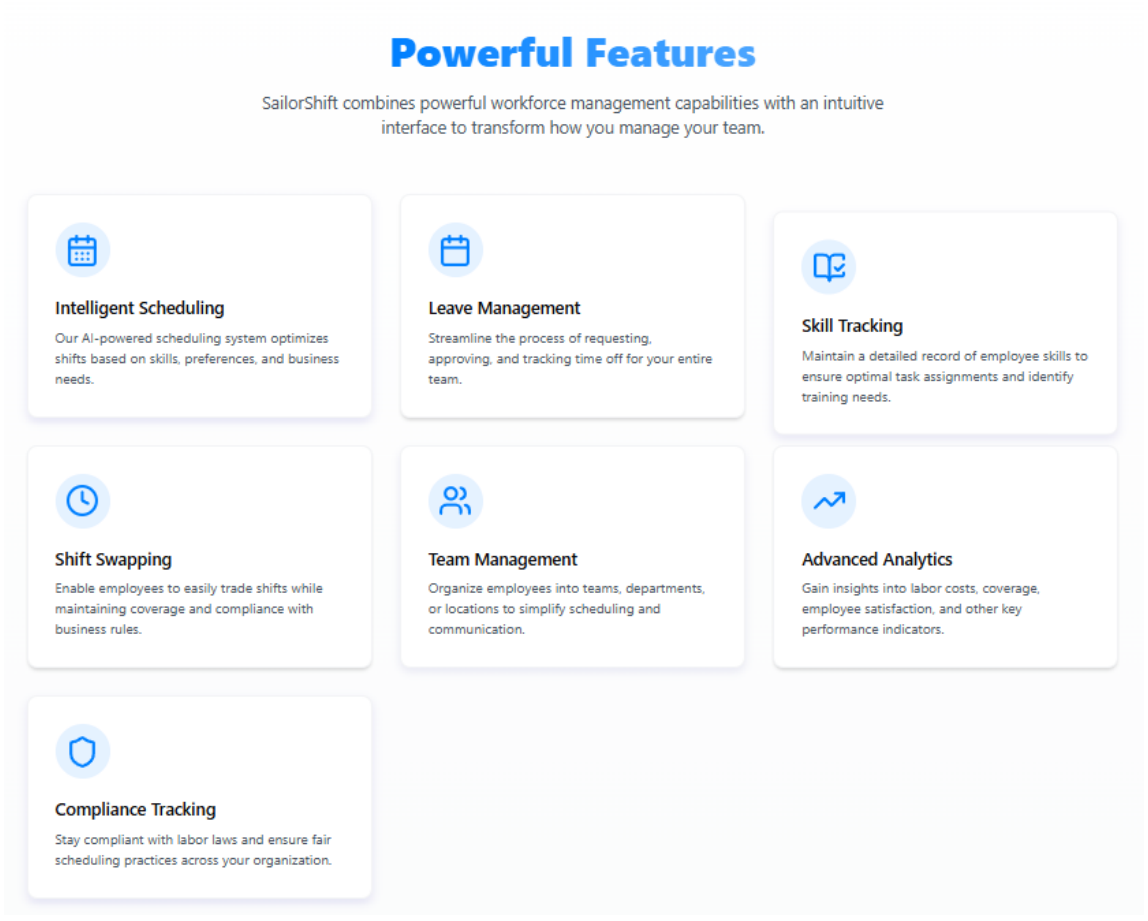
We're constantly learning from our users and evolving our platform. We embrace feedback and use it to build features that solve real problems.

Accessibility

We're committed to building tools that work for everyone. Our platform is designed to be accessible and user-friendly for people of all abilities.

Ready to Transform Your Workforce Management?

Join thousands of organizations already using SailorShift to create happier teams and more efficient operations.



Powerful Features

SailorShift combines powerful workforce management capabilities with an intuitive interface to transform how you manage your team.

Intelligent Scheduling

Our AI-powered scheduling system optimizes shifts based on skills, preferences, and business needs.

Leave Management

Streamline the process of requesting, approving, and tracking time off for your entire team.

Skill Tracking

Maintain a detailed record of employee skills to ensure optimal task assignments and identify training needs.

Shift Swapping

Enable employees to easily trade shifts while maintaining coverage and compliance with business rules.

Team Management

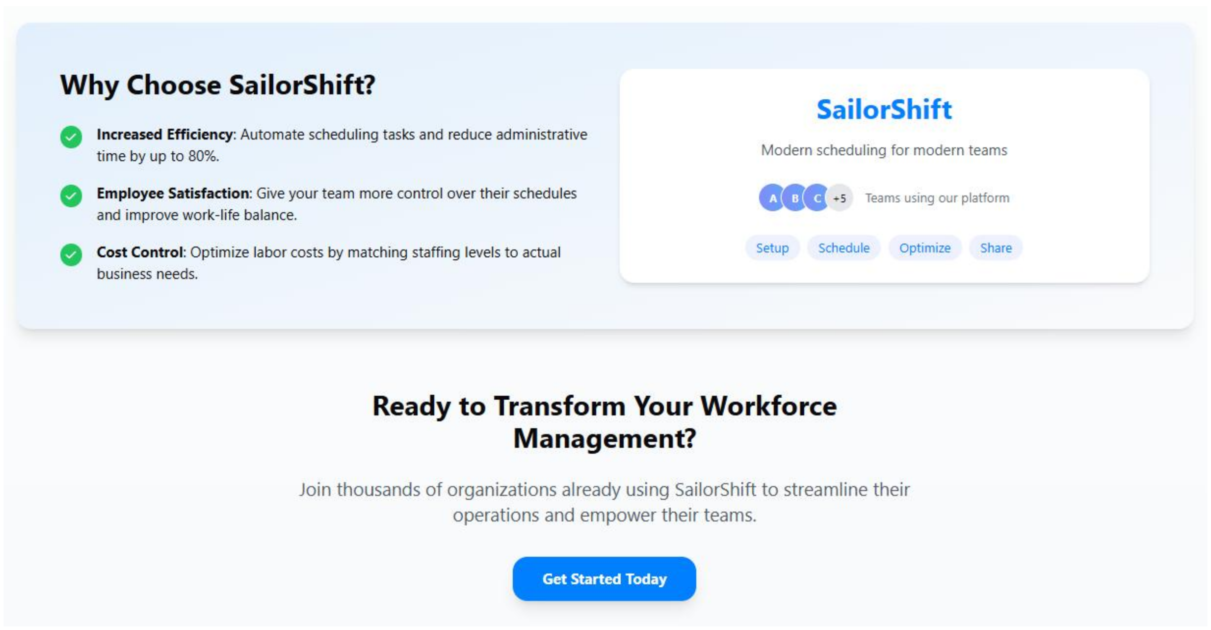
Organize employees into teams, departments, or locations to simplify scheduling and communication.

Advanced Analytics

Gain insights into labor costs, coverage, employee satisfaction, and other key performance indicators.

Compliance Tracking

Stay compliant with labor laws and ensure fair scheduling practices across your organization.



Why Choose SailorShift?

- **Increased Efficiency**: Automate scheduling tasks and reduce administrative time by up to 80%.
- **Employee Satisfaction**: Give your team more control over their schedules and improve work-life balance.
- **Cost Control**: Optimize labor costs by matching staffing levels to actual business needs.

SailorShift

Modern scheduling for modern teams

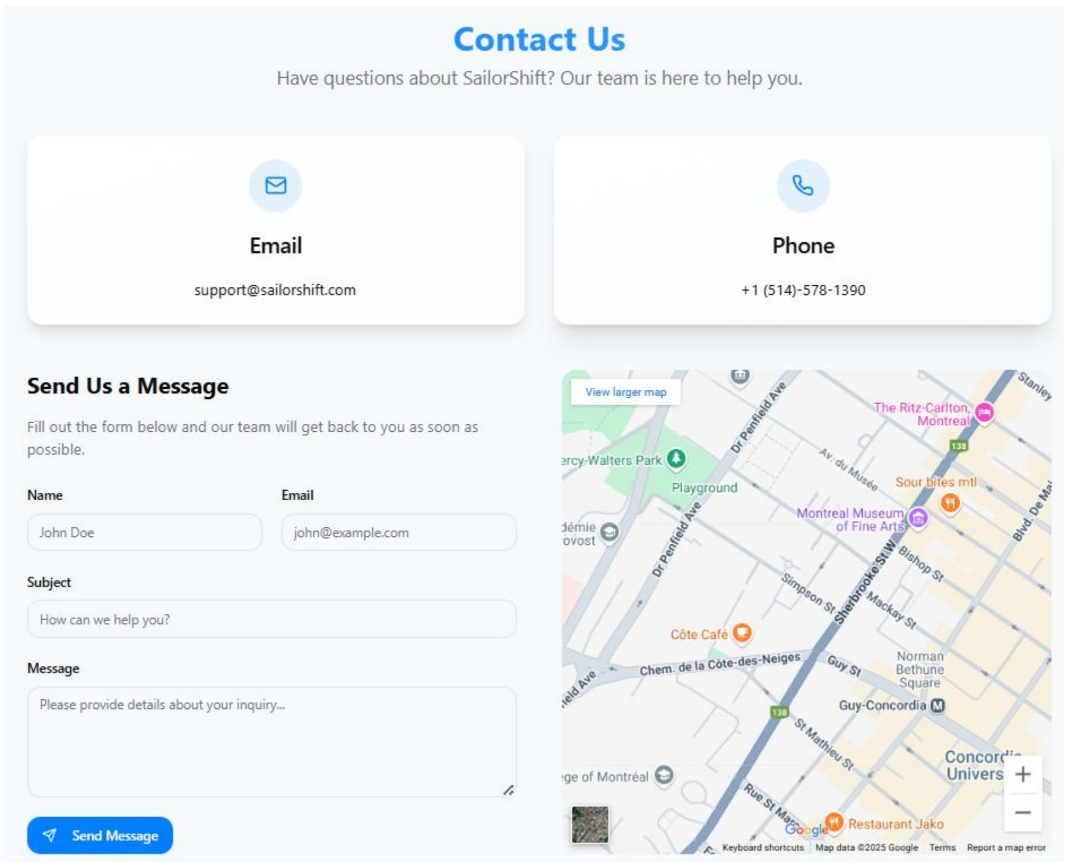
A B C +5 Teams using our platform

Setup Schedule Optimize Share

Ready to Transform Your Workforce Management?

Join thousands of organizations already using SailorShift to streamline their operations and empower their teams.

Get Started Today



Contact Us

Have questions about SailorShift? Our team is here to help you.

Email

support@sailorshift.com

Phone

+1 (514)-578-1390

Send Us a Message

Fill out the form below and our team will get back to you as soon as possible.

Name	Email
John Doe	john@example.com

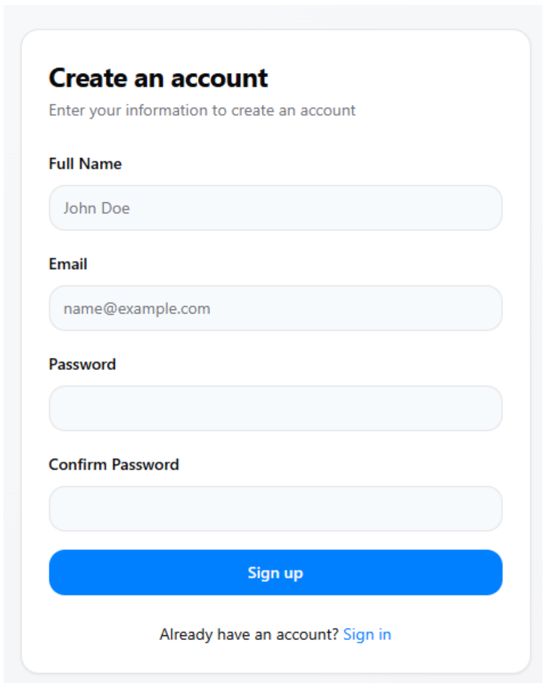
Subject

How can we help you?

Message

Please provide details about your inquiry...

Send Message



Create an account

Enter your information to create an account

Full Name

John Doe

Email

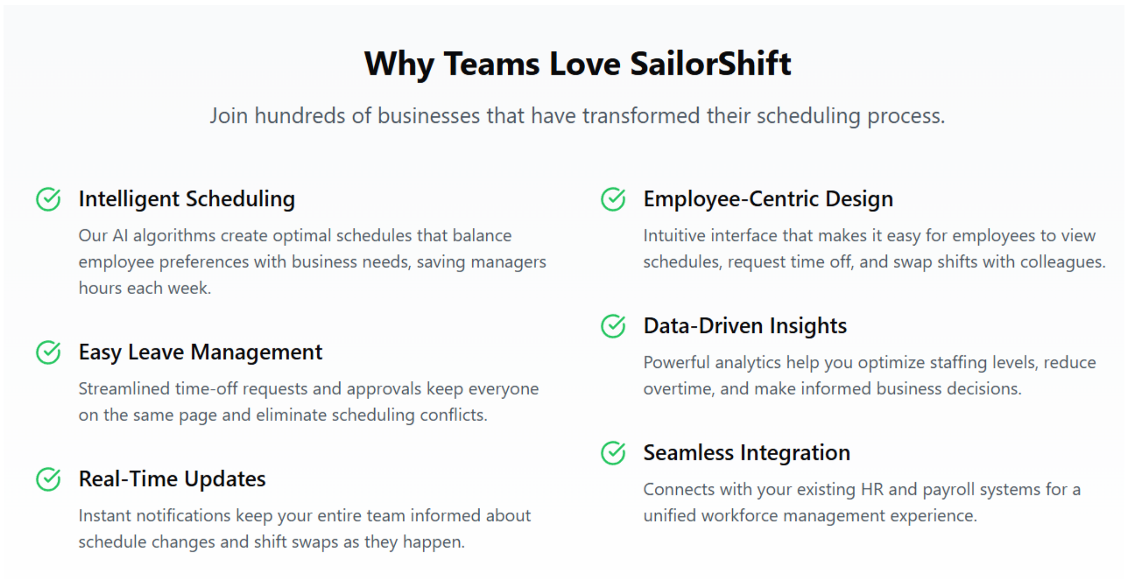
name@example.com

Password

Confirm Password

Sign up

Already have an account? Sign in



Why Teams Love SailorShift

Join hundreds of businesses that have transformed their scheduling process.

✓ Intelligent Scheduling

Our AI algorithms create optimal schedules that balance employee preferences with business needs, saving managers hours each week.

✓ Easy Leave Management

Streamlined time-off requests and approvals keep everyone on the same page and eliminate scheduling conflicts.

✓ Real-Time Updates

Instant notifications keep your entire team informed about schedule changes and shift swaps as they happen.

✓ Employee-Centric Design

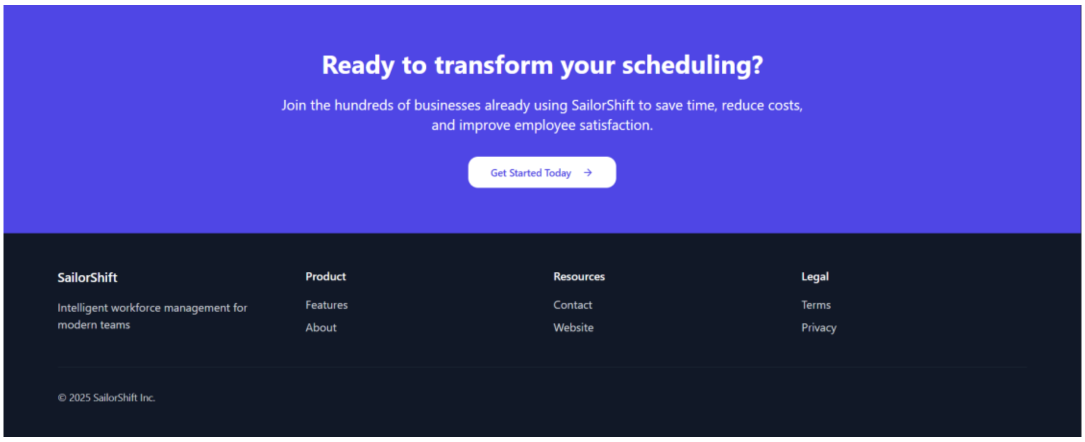
Intuitive interface that makes it easy for employees to view schedules, request time off, and swap shifts with colleagues.

✓ Data-Driven Insights

Powerful analytics help you optimize staffing levels, reduce overtime, and make informed business decisions.

✓ Seamless Integration

Connects with your existing HR and payroll systems for a unified workforce management experience.



Ready to transform your scheduling?

Join the hundreds of businesses already using SailorShift to save time, reduce costs, and improve employee satisfaction.

Get Started Today →

SailorShift

Intelligent workforce management for modern teams

Product

Features

About

Resources

Contact

Website

Legal

Terms

Privacy

© 2025 SailorShift Inc.

VII. Individual Contributions by each Team Member

1. **Sai Krishna Prashanth Kolluru** (Employee Website, Cloud Services Management, Deployment & User Updates Management System)

- Developed Employee Website (Front-end UI & Backend).
- Developed dedicated employee dashboard with all necessary pages for employee for submitting leave, swap requests and checking schedule and developed a shift swap algorithm which automatically reassigns shifts based on employee schedules for balanced workloads.
- Integrated Supabase database tables with front end for Cloud Database Management.
- Implemented an instant User updates management system (Account credentials, password recovery, leave request updates, swap request updates through email notification system).
- Managed user authentication and user access control for websites.
- Report Writing and Project Documentation.

2. **Saisri Durga Shanmukha Abhiram Kalvakolanu** (Employer Website, AI Chatbot, Database Management & Leave Requests, Swap Requests and Overall Optimization)

- Developed Employer Website (Front-end & Back-end) using React, Flask, HTML, CSS, Node JS etc.
- Designed and Implemented Leave Request, Shift swap, Schedule Optimization and Preferences Algorithms for automated approvals using Skill ratings.
- Designed and Managed PostgreSQL Database Tables and linked the database to Python codes to create, update and delete tables time to time.
- Designed and Integrated AI-Powered Chatbot using Gemini 2.0 Flash in Employee portal with RAG functionality.
- Developed Flask App to launch the website for development and testing.
- Developed Chatbot backend logic and integrated it with Leave, Swap and Preferences requests.
- Report Writing and Project Documentation.

3. **Avinash Pothabattula** (System Optimization, Structure & Scheduling Logic)

- Integrated Scheduling Optimization Algorithm using PuLP library in Python.
- Worked on Workforce Balancing & Shift Swap Optimization.
- Worked with Google Colab to develop Scheduling logic.
- Contributed to the development of ideas and helped design the structure and flow of various components within the project.
- Designed poster for poster presentation.

4. **Roshan Kotapati** (Testing, Scheduling, & Optimization Assistance)

- Assisted in System Optimization & Scheduling Logic Implementation.
- Conducted Unit, Integration & User Acceptance Testing.
- Identified & fixed bugs to improve performance.
- Ensured smooth user experience through UI testing & refinements.
- Conducted Functional, Unit, and Integration Testing.

5. **Lalith Chand Saripilli** (Testing & Maintenance)

- Collaborated on fixing performance issues & debugging.
- Spoke with various stores for information and data.
- Ensured all technical documentation and reports were well-structured.