

YYC Food Trucks Scheduler

Kim, Yoohyun 000928855

Motyer, Leona 000931148

Pratap, Rudra 000935797

Singh, Tehjib 000934598

Laraya, Chester 000847615

Table Of Contents

Title Page ............................................................................................................... 1

Table of contents ................................................................................................... 2

Project identification .............................................................................................. 3-4

1.1 Project title

1.2 External client information

1.3 Client current system

1.4 Proposed solution

Use case diagram Both(phase1&2).................................................................... 5-6

Use case descriptions ........................................................................................... 6-8

Class diagram (Phase 1& 2) .................................................................................... 9-10

ERD Dragram ........................................................................................................ 11

Deployment Diagram ............................................................................................ 12

Database Reasoning ............................................................................................... 13

Figma UI ................................................................................................................. 14-26

Mutual Confidentiality Agreement ……………………………………………. 27

Team Contract ..................................................................................................28-33

Workload Table .................................................................................................... 33-34

**Project Identification**

**1.1 Project Title**

YYC Food Trucks Scheduler

1.2 External Client Information

**Client Name:** Jennifer Andrews   
 **Position:** Owner/Operator, YYC Food Trucks   
 **Contact Details:** 403-605-8806

**Organization Information:**

* **Type of Business:** Food truck event management company
* **Location:** Calgary, Alberta

**1.3 Sponsor’s Current System**

The client manually schedules events and assigns employees using paper and pen. This process is time-consuming, inefficient, and prone to errors.

**1.4 Proposed Solution**

Our team proposes developing a **Food Truck Event Scheduling System** that will:

* Allow the creation, editing, and deletion of events.
* Display event details, including location, date, time, and staff assignments.
* Manage employee records, including wage, availability, and location.
* Provide a scheduling interface that simplifies employee assignments based on availability, proximity, and cost-effectiveness.
* Offer a **week-at-a-glance** dashboard for event scheduling.
* Implement a **distance optimization algorithm** to suggest efficient assignments.

Our solution will help schedule by displaying employee availability and best assignments, making the process more efficient.

Current system

A notebook with writing on it

AI-generated content may be incorrect.

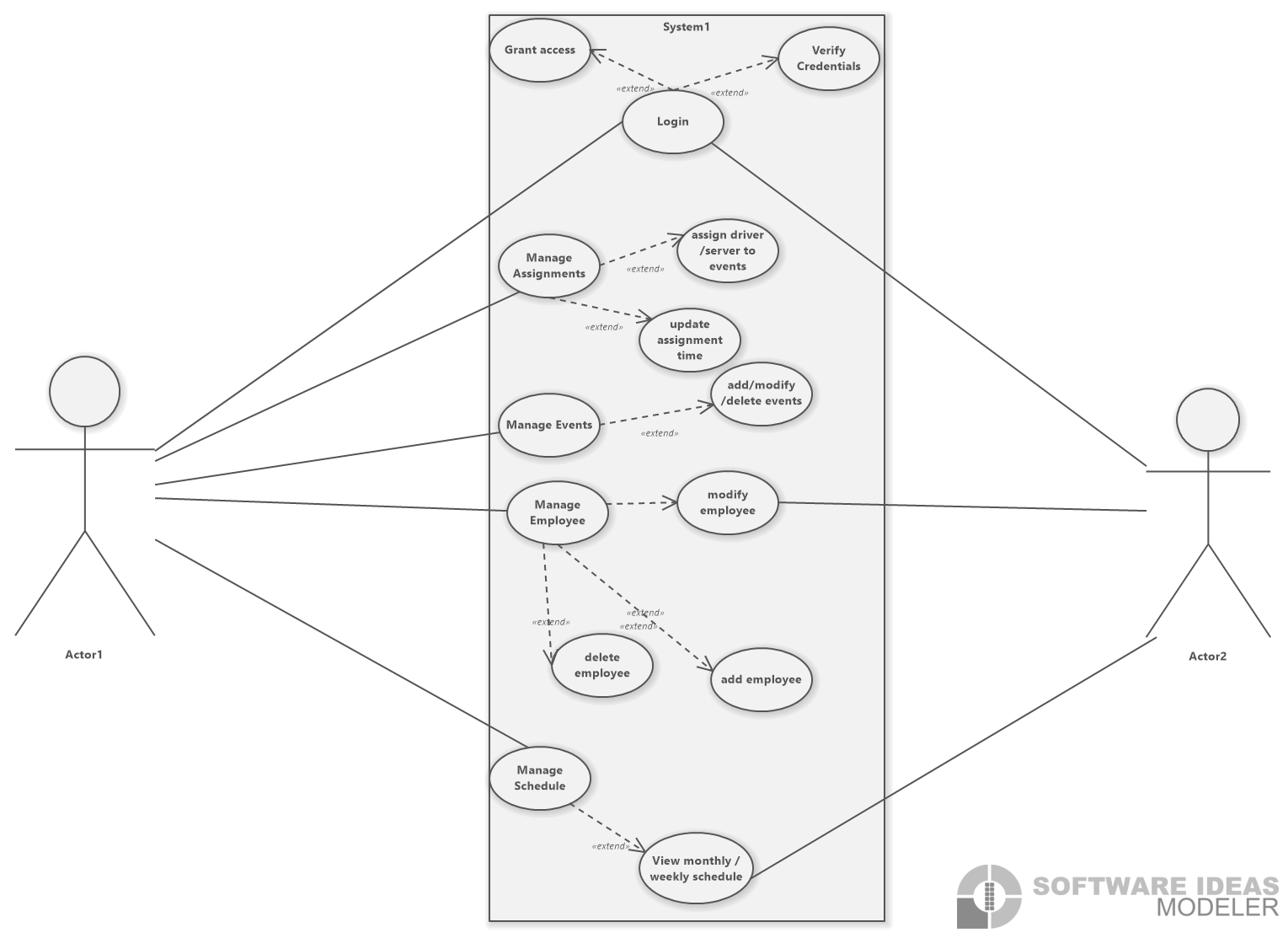
Use Case Diagram

PHASE !:

A diagram of a company

Description automatically generated

PHASE 2:



**1. Manage Employee**   
 **Description:**   
 The Administrator can manage employee records, including adding new employees, editing existing employee details, and removing employees from the system. Employee records include name, wage, availability, and residential location.

**Preconditions:**

* The Administrator is logged into the system.

**Postconditions:**

* The employee database is updated with changes.
* Changes to employee availability and details are reflected in the scheduling.

**Main Flow:**

1. The Administrator selects "Manage Employee."
2. The system displays options to add, change, or remove employee records.
3. The Administrator chooses the desired action:

* **Add Employee:** Enters details and sends.
* **Modify Employee:** Selects an existing record, updates fields, and sends.
* **Delete Employee:** Selects a record and confirms deletion.

1. The system confirms input and updates the database.

**2. Manage Events**   
 **Description:**   
 The Administrator can create, edit, and remove events in the system. Events include location, date, time, and assigned staff.

**Preconditions:**

* The Administrator is logged into the system.

**Postconditions:**

* The event schedule is updated with the latest information.

**Main Flow:**

1. The Administrator selects "Manage Events."
2. The system displays options to add, change, or remove events.
3. The Administrator chooses the desired action:

* **Add Event:** Enters event details, assigns staff, and sends.
* **Modify Event:** Selects an event, updates fields, and sends.
* **Delete Event:** Selects an event and confirms deletion.

1. The system confirms input and updates the event schedule.

**3. Assign Driver/Server to Events**

**Description:**   
 The Administrator assigns drivers and servers to events based on proximity, availability, and wage.

**Preconditions:**

* The Administrator is logged into the system.
* Employee and event records exist in the system.

**Postconditions:**

* Staff assignments are updated for the selected event.
* Assignment data is available for review in the weekly schedule.

**Main Flow:**

1. The Administrator selects "Manage Schedule" and chooses an event.
2. The system displays available employees based on scheduling constraints.
3. The Administrator selects drivers and servers to assign to the event.
4. The system calculates and refines assignments for efficiency.
5. The Administrator confirms the assignment.

Shape**4. View Weekly/Monthly Schedule**

**Description:**   
 The Owner/Operator and Administrator can view a summarized schedule of events in a weekly or monthly format, including assigned staff, locations, and dates.

**Preconditions:**

* The user is logged into the system.
* Events have been scheduled.

**Postconditions:**

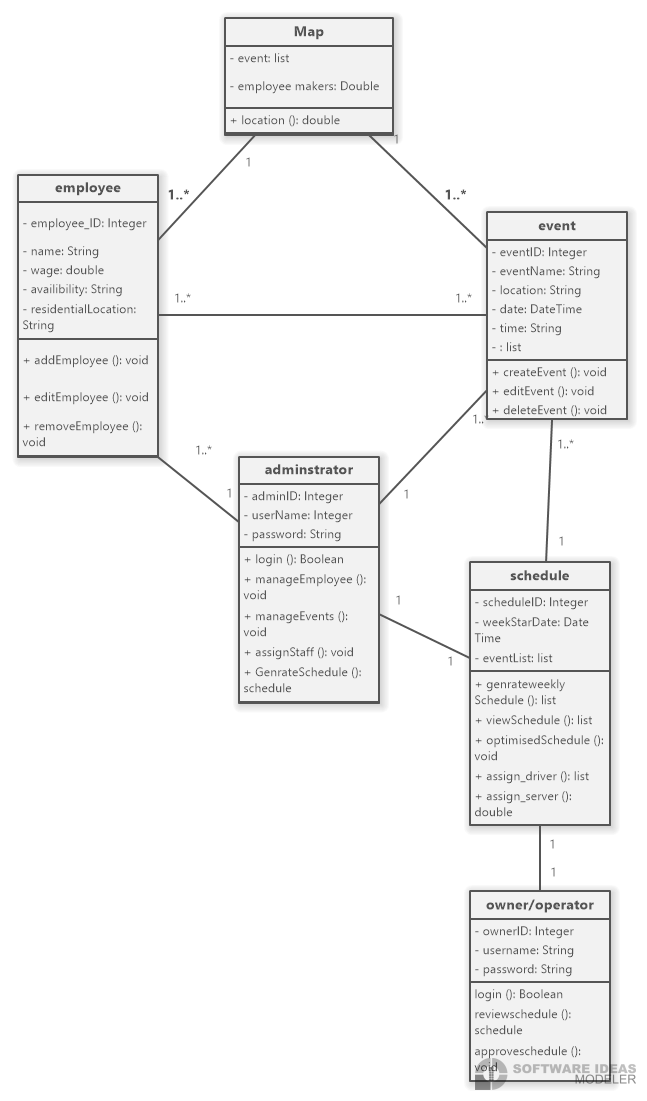
* A clear, printable view of the schedule is displayed.

**Main Flow:**

1. The user selects "View Schedule."
2. The system retrieves all events for the selected time (weekly/monthly).
3. The schedule is displayed, showing event details, and assigned staff.
4. The user can filter, or sort events based on criteria (e.g., date or location).

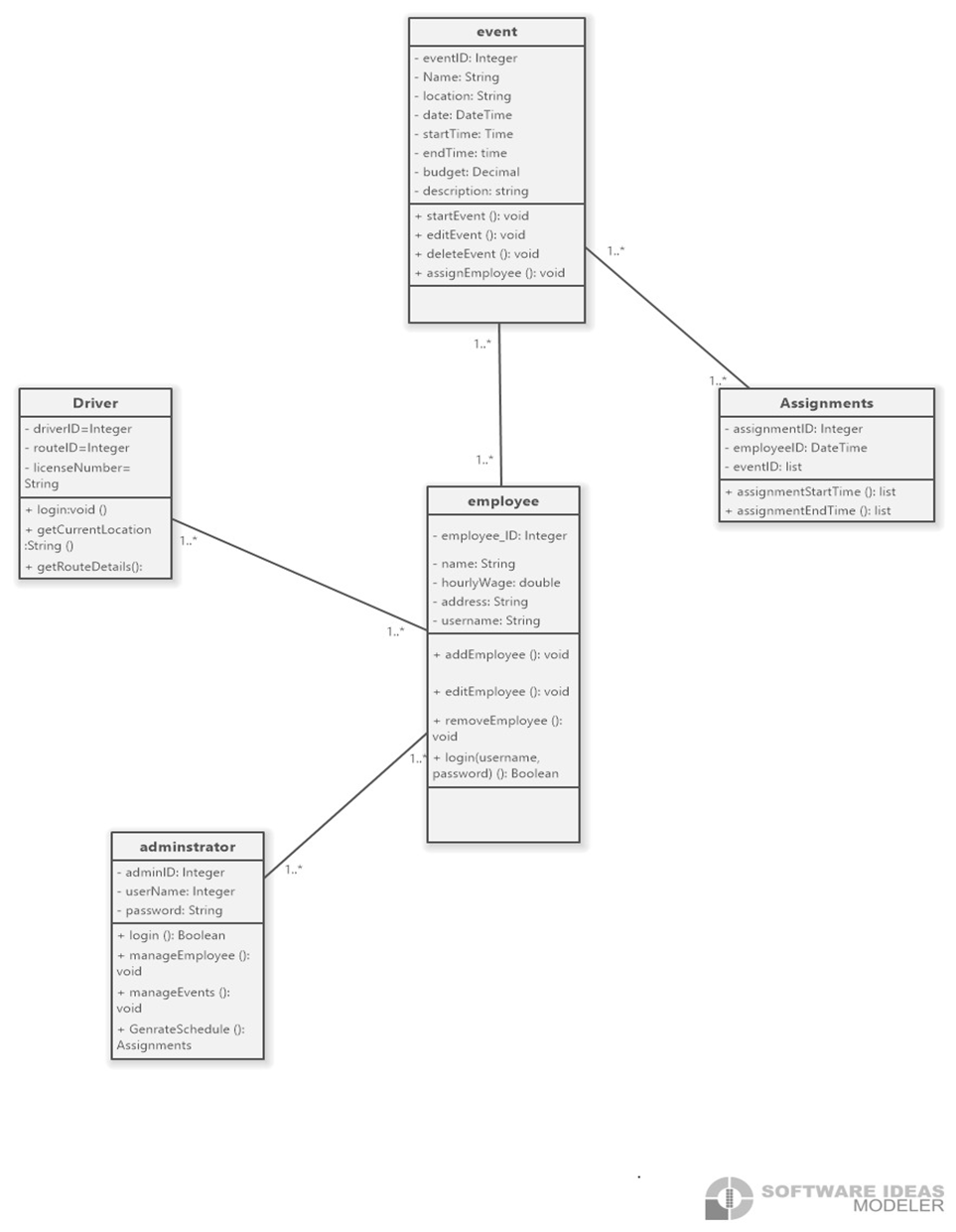
**Phase 1:**

Class Diagram



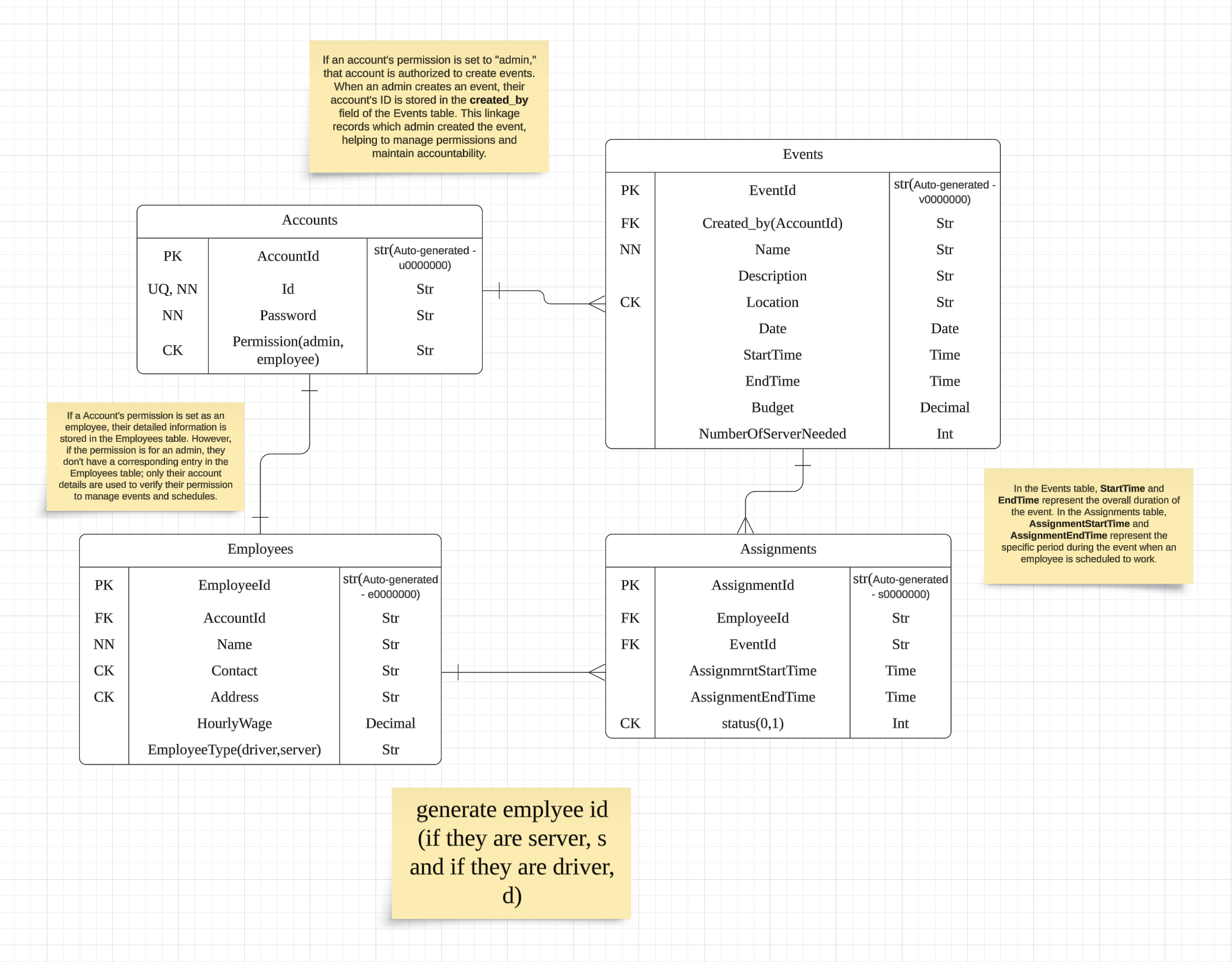
**Phase 2:**

Class Diagram



# ERD Diagram

[Food Truck app: Lucidchart](https://lucid.app/lucidchart/63424edf-3f00-4462-a163-71b560e3068b/edit?invitationId=inv_8af893ac-7761-4b91-a347-3582cb8f14d9&page=0_0#)



# Deployment Diagram

A screenshot of a computer

AI-generated content may be incorrect.

# DATABASE

PostgreSQL

**Reasoning for Choosing PostgreSQL for Our Employee Scheduling Application**

Our team has decided to use **PostgreSQL** as our employee scheduling application database due to its reliability, efficiency, and ability to handle complex data structures. Given the nature of our project, which involves assigning shifts, tracking availability, and preventing scheduling conflicts, we required a robust **relational database management system (RDBMS)** that ensures data accuracy and consistency.

One of the primary reasons we selected PostgreSQL is its **ability to handle complex queries and joins efficiently**. Since our application must manage employee schedules across various departments, PostgreSQL's structured query capabilities allow for seamless data retrieval and organization. Moreover, **its built-in support for JSON** provides flexibility in storing semi-structured data, such as employee preferences and shift histories, without requiring extensive schema modifications.

Another crucial factor in our decision is PostgreSQL's **strong concurrency management**. The system ensures **multi-user support**, preventing data conflicts when multiple managers update schedules simultaneously. This feature is essential in an environment where real-time updates are necessary to maintain operational efficiency. Additionally, PostgreSQL is **highly scalable**, which allows our application to expand without significant performance degradation as our user base grows.

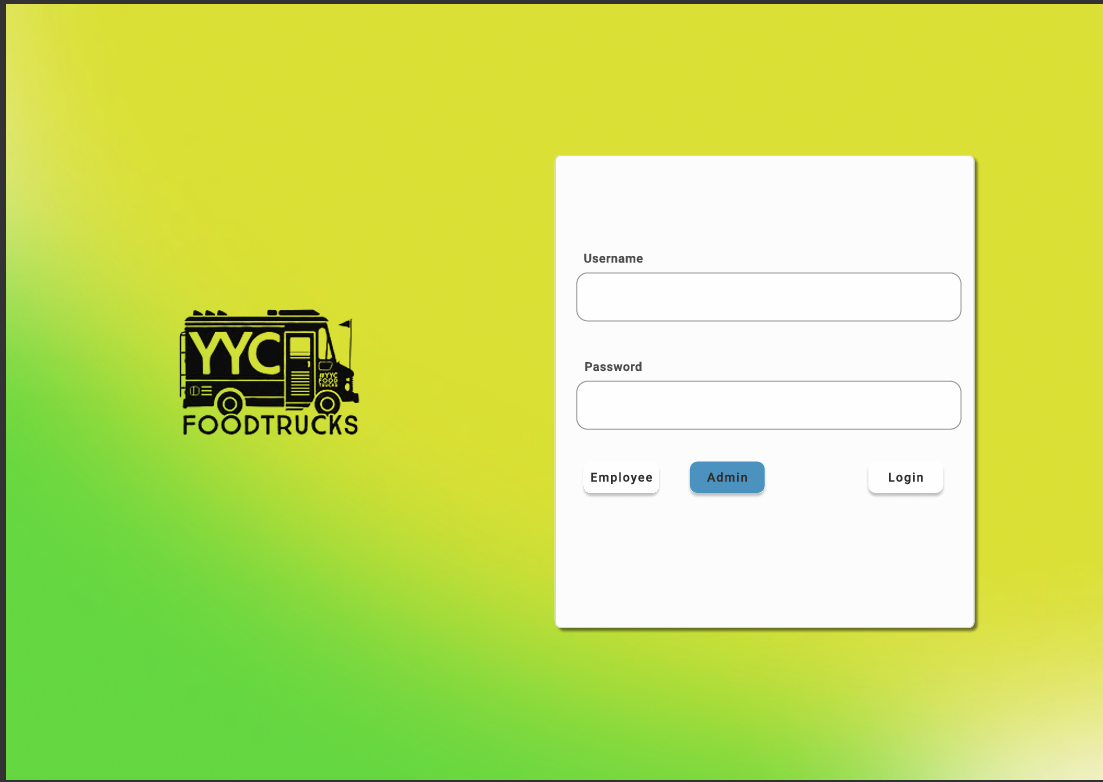
Security considerations also played a pivotal role in our database selection. PostgreSQL provides **role-based access control (RBAC)**, allowing us to **protect sensitive employee data** by restricting access based on user roles. Furthermore, PostgreSQL offers **advanced extensions**, such as automation tools, which can be leveraged to enhance functionality, including sending shift reminders and generating reports.

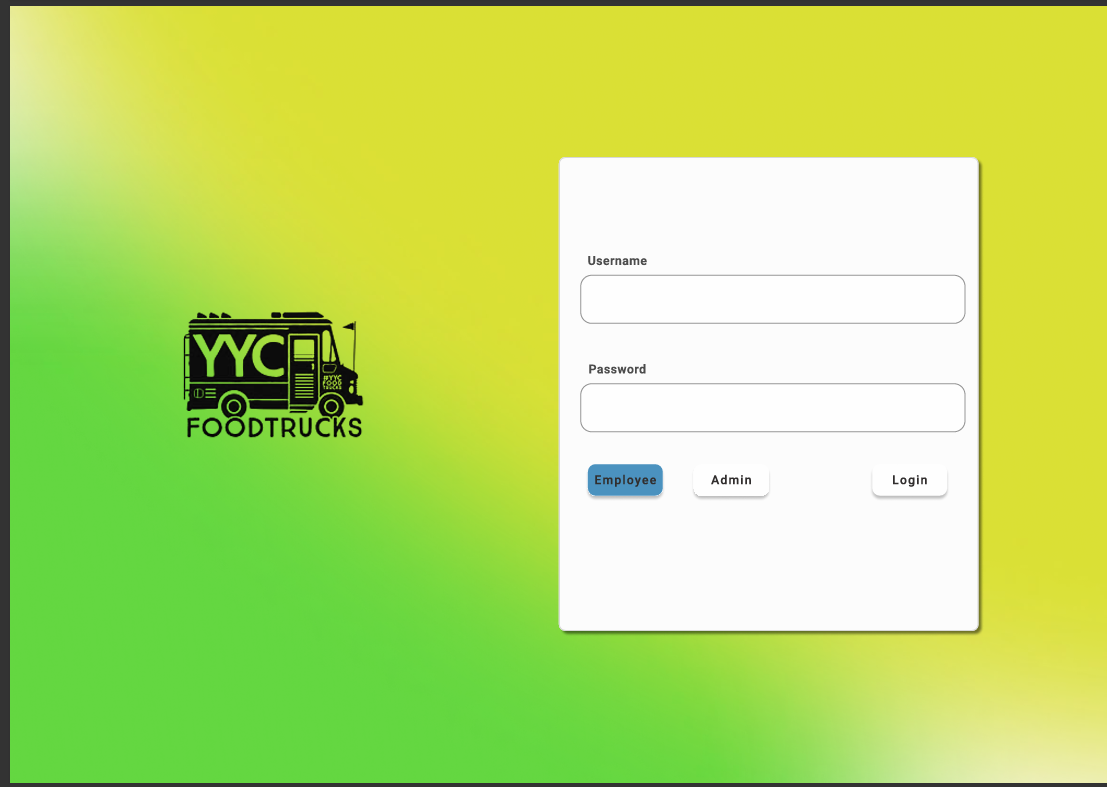
PostgreSQL is superior to MySQL due to its **richer feature set**, including **built-in publish/subscribe notifications** that improve system communication. Additionally, PostgreSQL is widely adopted in the industry, with major cloud providers such as **AWS offering PostgreSQL support through RDS**. Kinsella also emphasizes that learning an **RDBMS like PostgreSQL provides a deeper understanding of schemas, relationships, and constraints**, which is fundamental for efficient application data management.

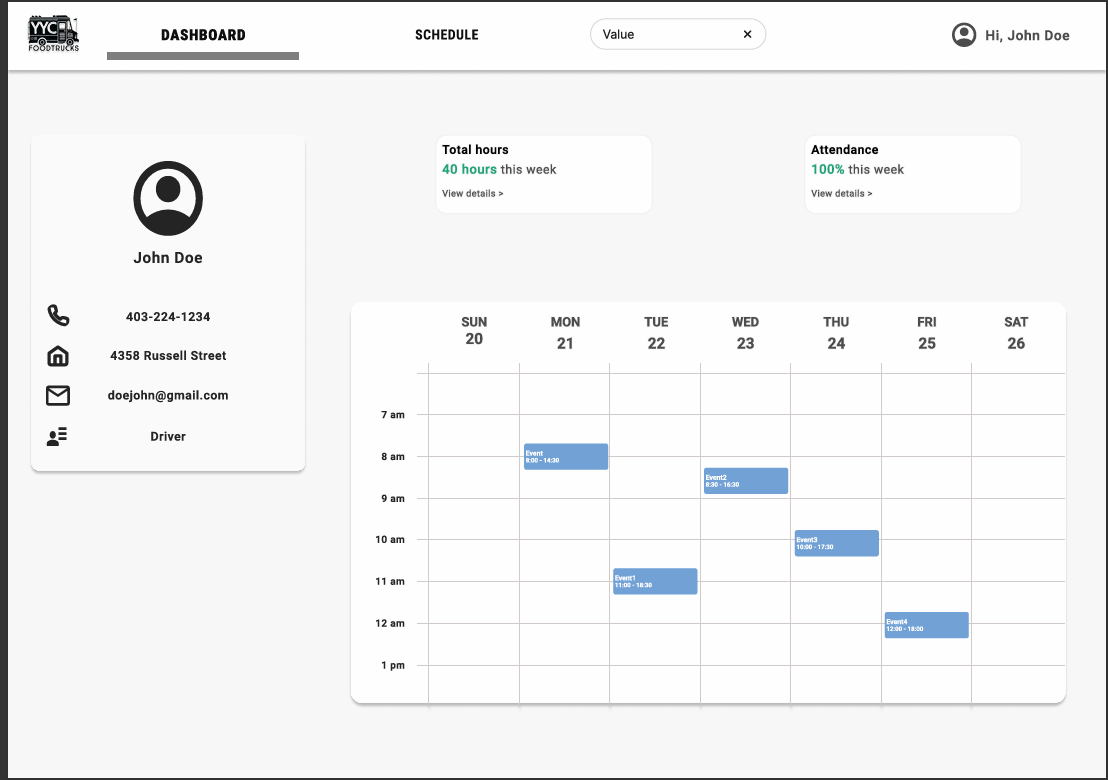
In conclusion, PostgreSQL offers the **reliability, flexibility, scalability, and security** required to build a future-proof employee scheduling application. Its ability to handle **complex queries, concurrent users, and structured data relationships** makes it the optimal choice for our project.

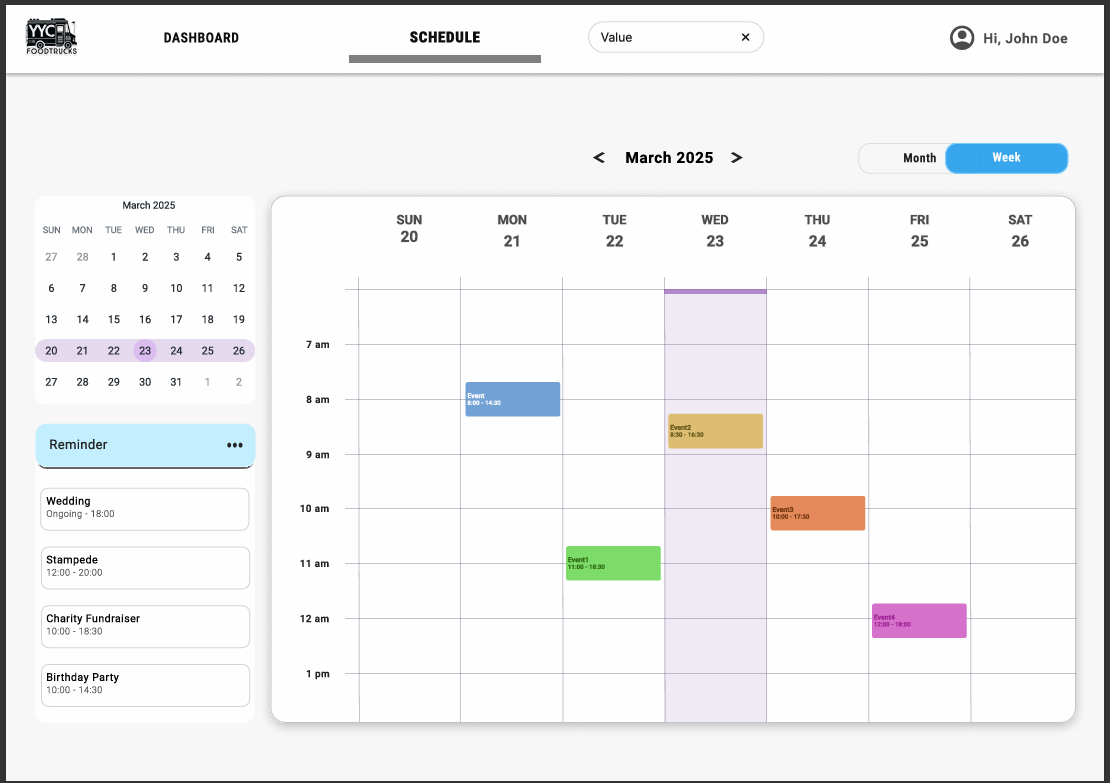
Figma UI

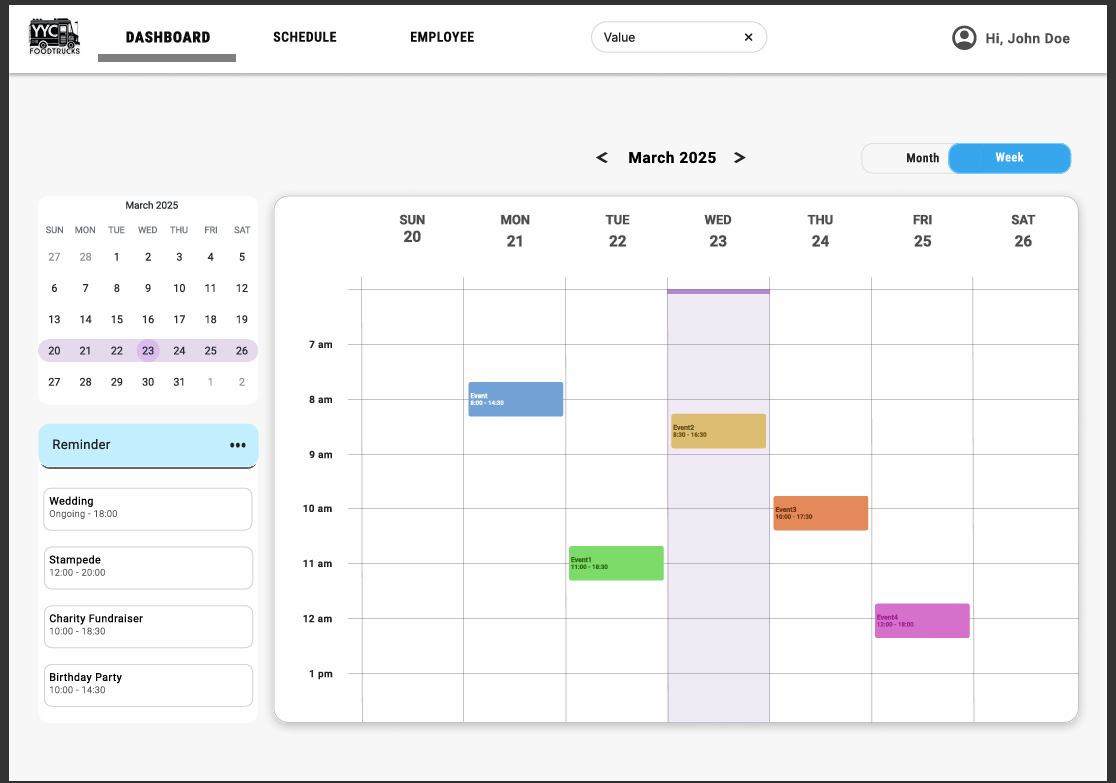
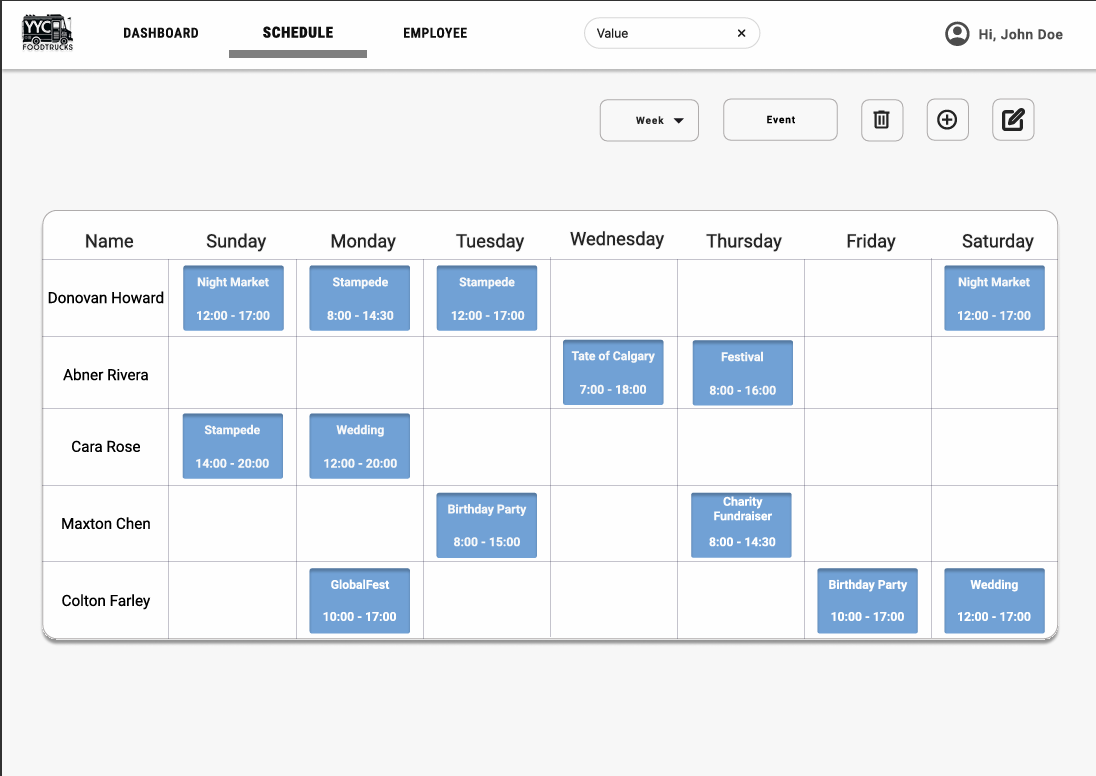
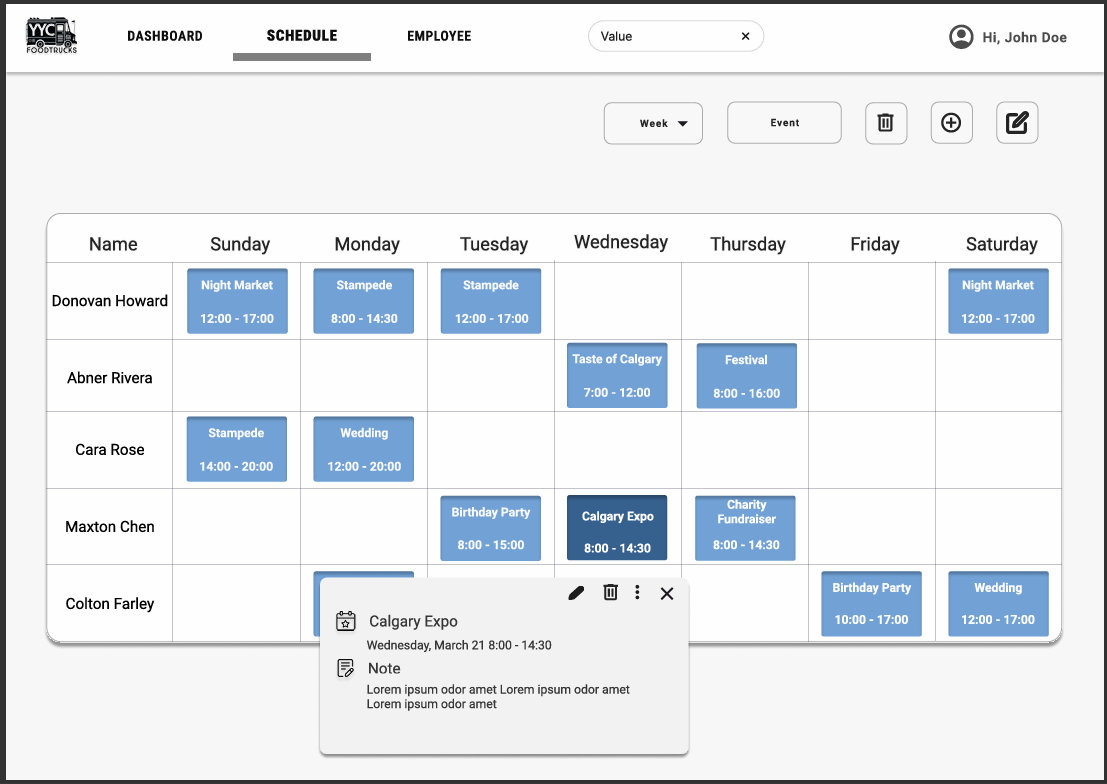
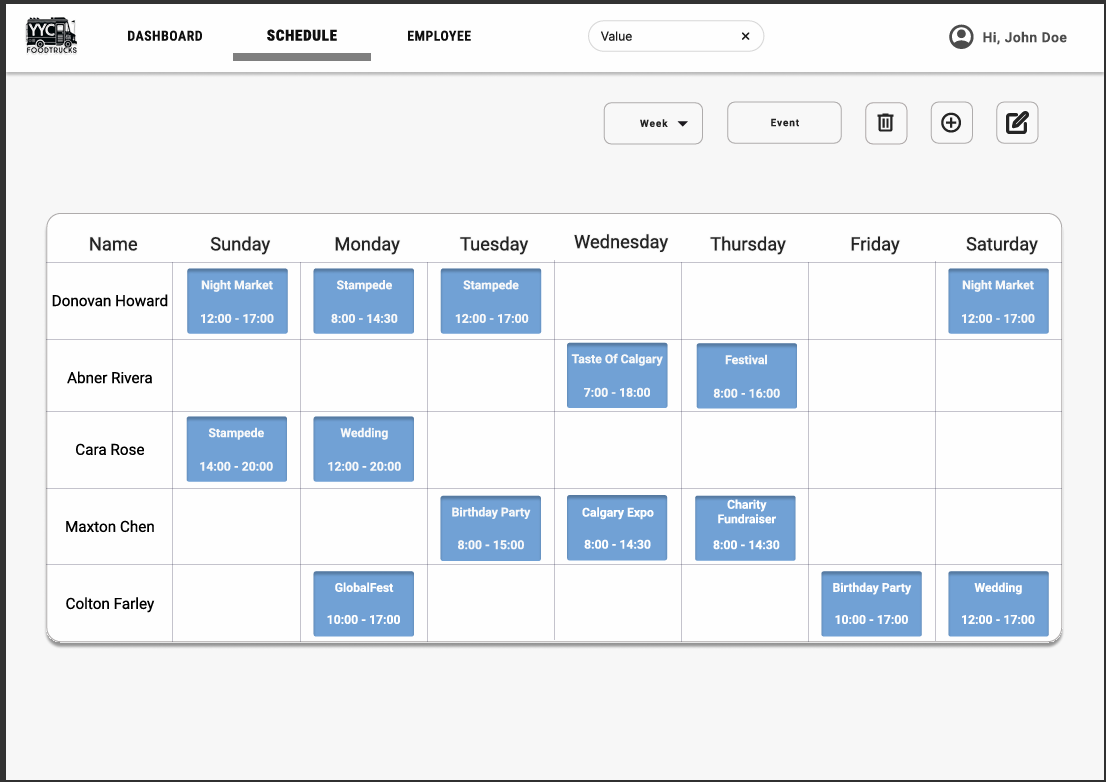
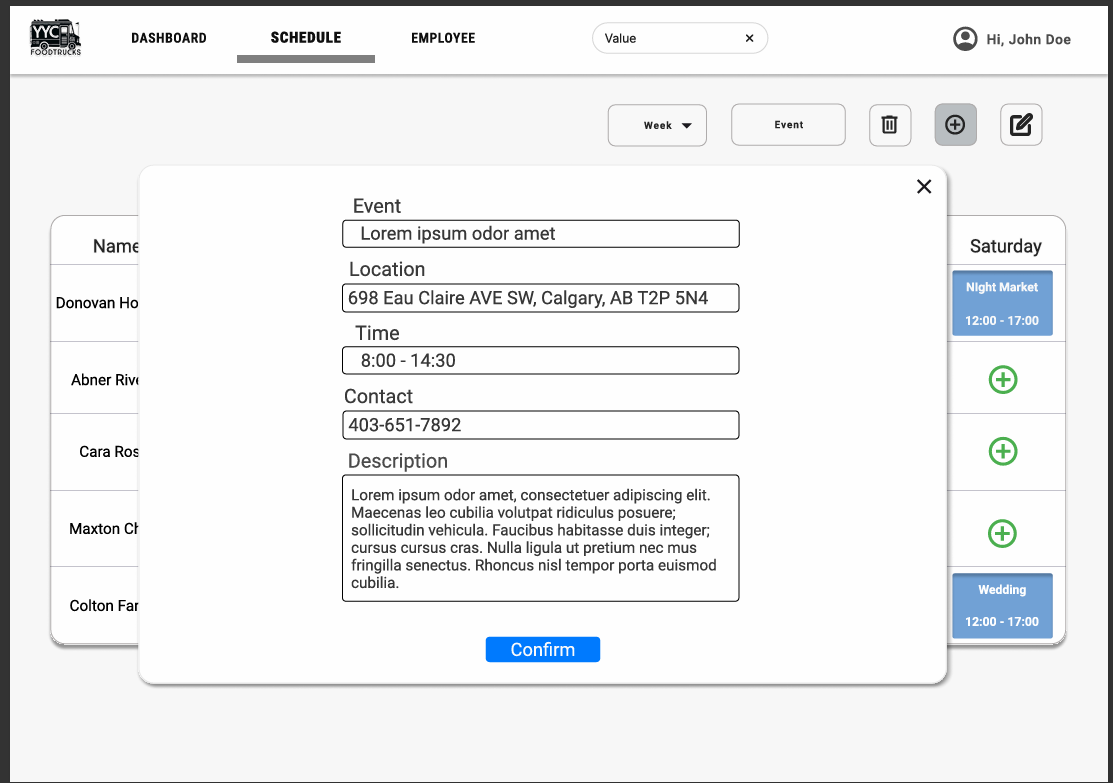
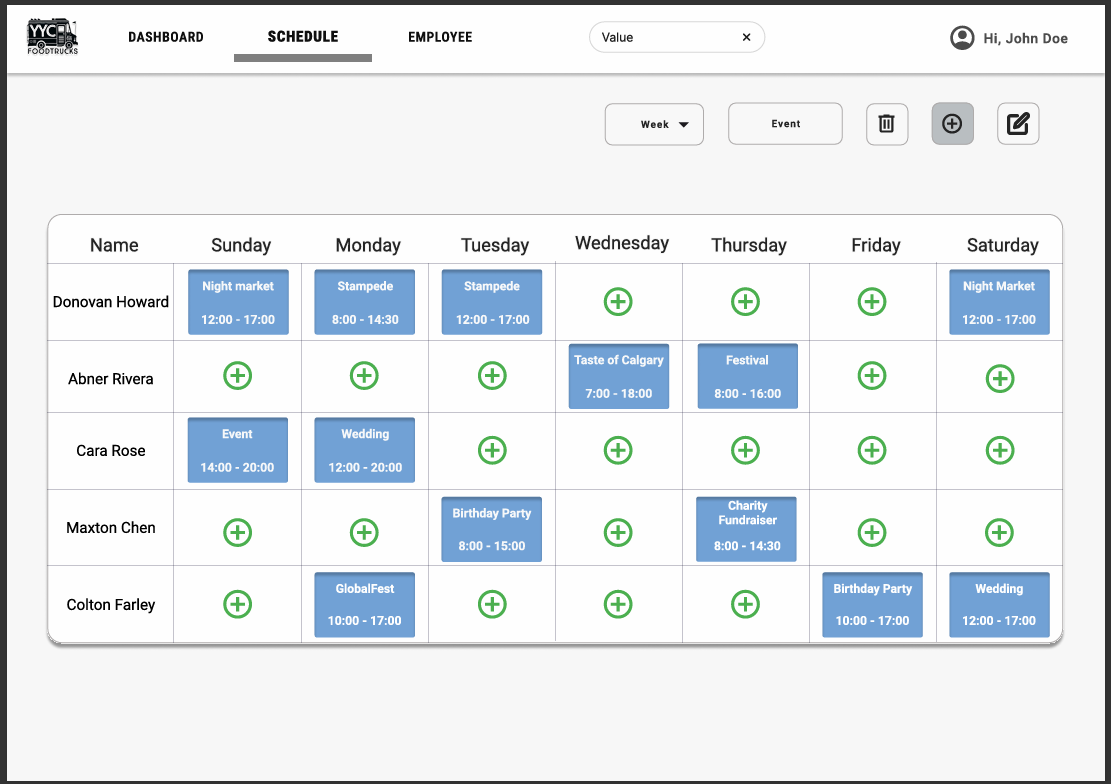
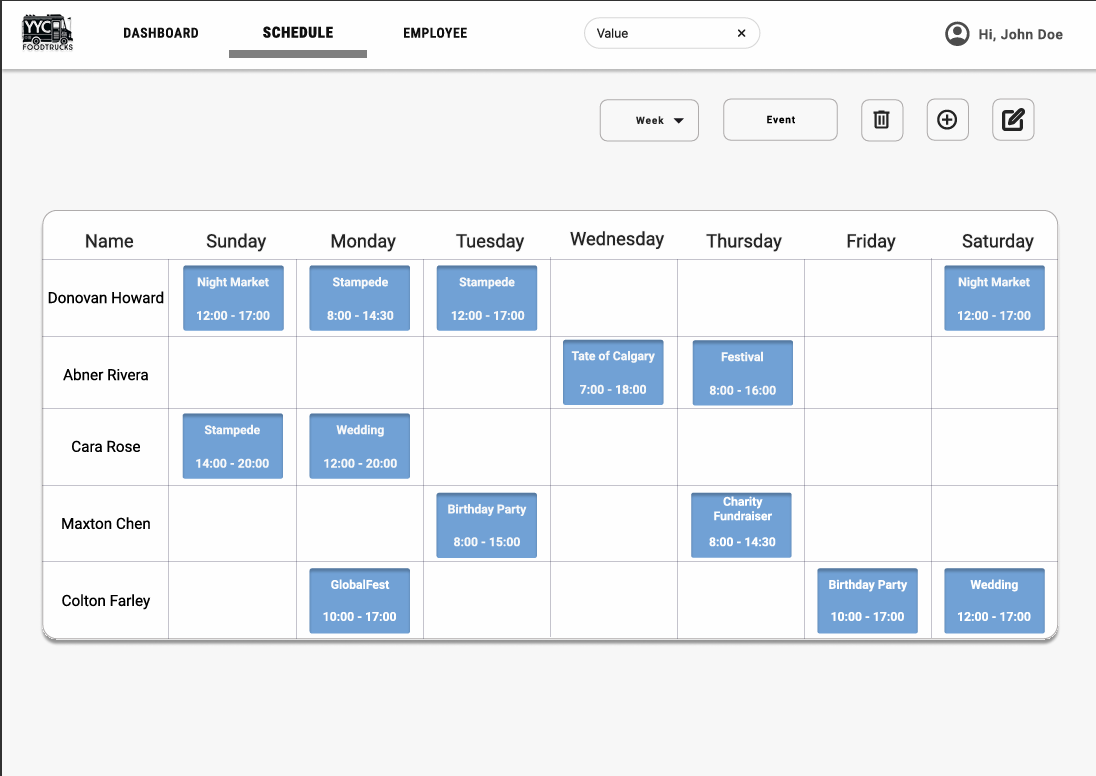
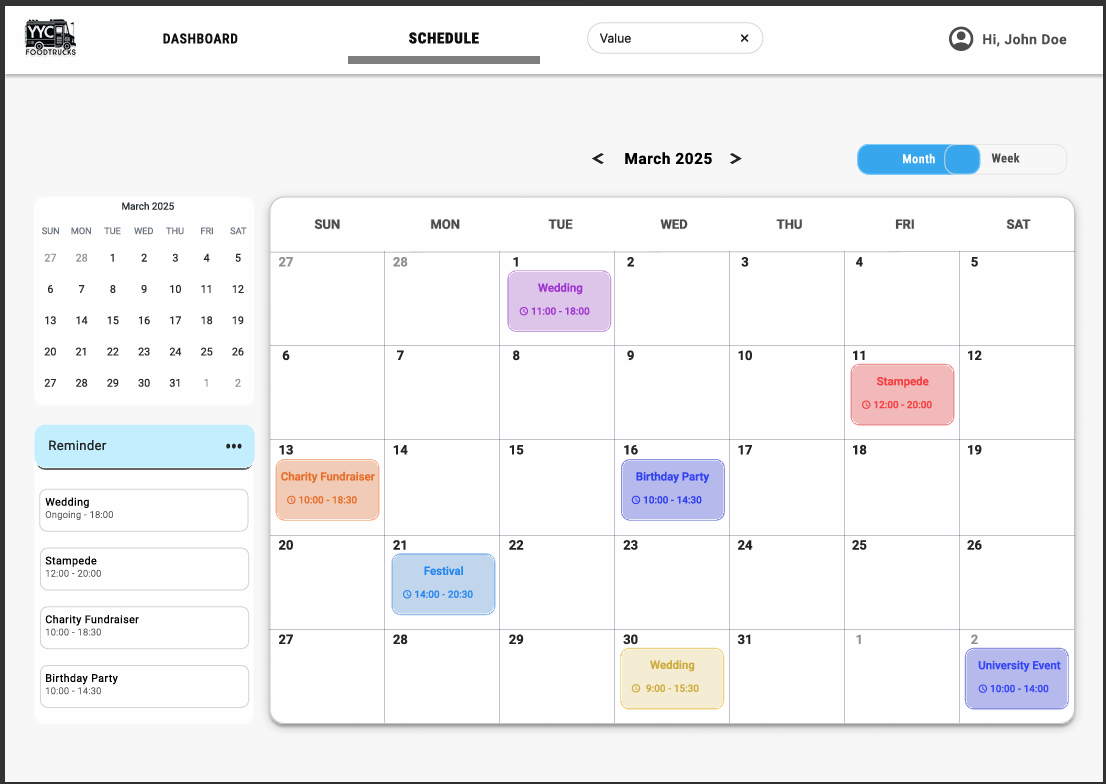
[figma link](https://www.figma.com/proto/HMDQ3ywBDrNQ25gT7fc4wF/cpsy301?node-id=0-1&t=c9Xyha5AkX0yvDwb-1)

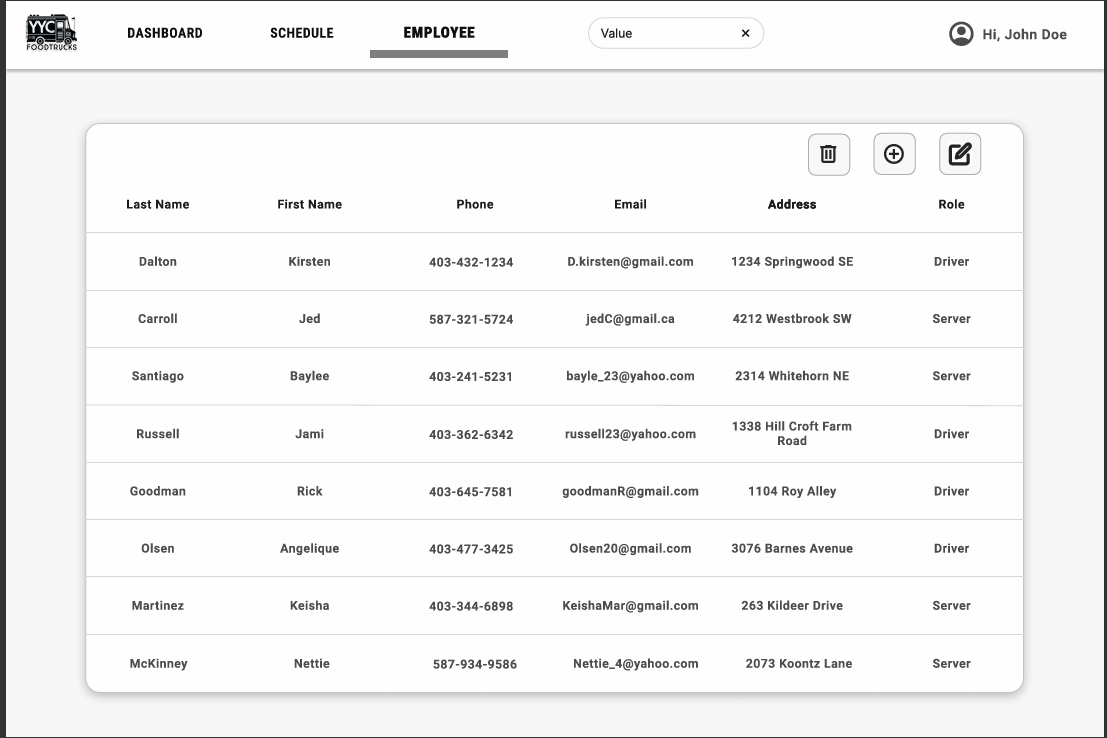
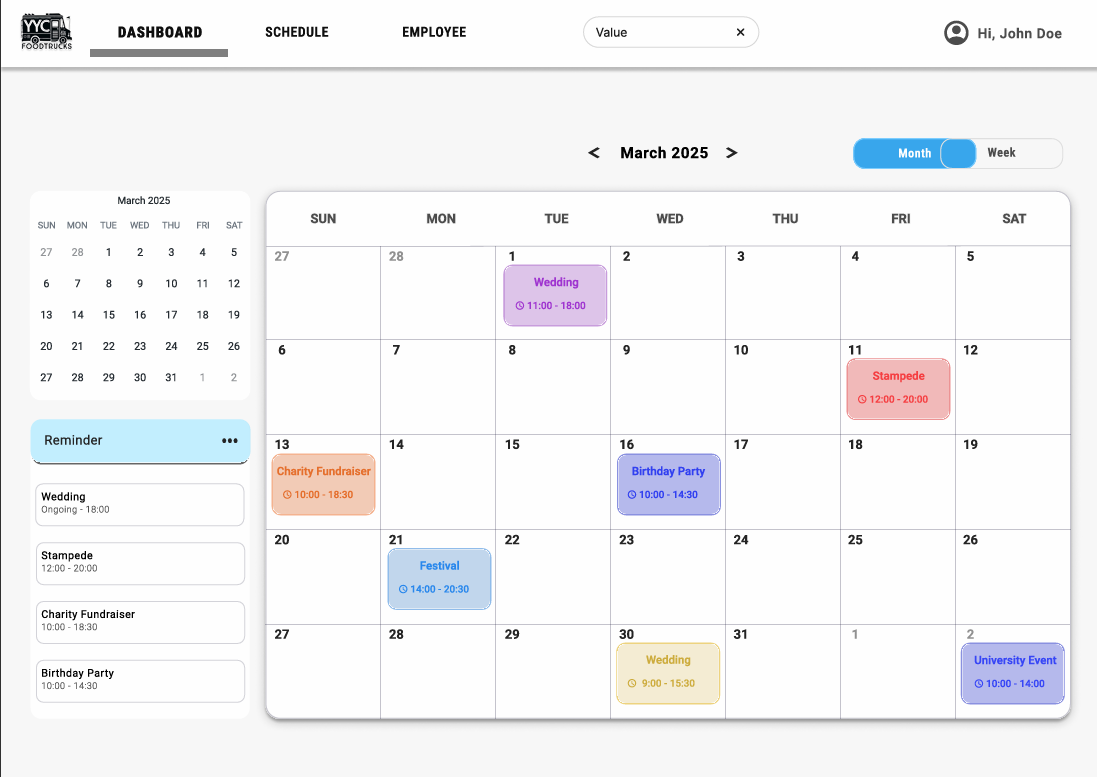












**Mutual Confidentiality Agreement**

**Effective Date**: Jan 6, 2025

**Parties**: This Agreement is made between The GOATS ("Party A") and YYC Food Trucks ("Party B").

**Purpose**: Both parties intend to engage in discussions and activities related to YYC Food Trucks Scheduler("Project"), during which confidential information may be exchanged.

**Confidential Information**: For this Agreement, "Confidential Information" includes all non-public information disclosed by either party, directly or indirectly, in writing, orally, or by inspection of tangible objects, that is designated as confidential or that reasonably should be understood to be confidential given the nature of the information and the circumstances of disclosure.

**Obligations**: Party A and Party B agree not to show or use each other's Confidential Information for any purpose outside the scope of the Project, except with the prior written consent of the disclosing party.

**Duration**: This Agreement shall remain in effect for one year from the Effective Date or until the Confidential Information no longer qualifies as confidential, whichever occurs first.

**Miscellaneous**: This Agreement constitutes the entire understanding between the parties about confidentiality and supersedes all prior discussions and agreements. Any amendments must be made in writing and signed by both parties.

**Team Contract**

**Team Name:** The GOATS

**Team Crest**

A drawing of a goat with horns

AI-generated content may be incorrect.

**Team Members and Professional Biographies**

**Leona Motyer:**    
 Software dev with a soft spot for infrastructure engineering. Leona’s skills include personality hire, creating naming schemas that are funny, meeting deadlines (most of the time), solving problems without breaking a sweat, and squashing bugs like it is a game. She has some solid internship experience and knows how to lead without acting like a bossy batch.

**Rudra:**    
 I am a code craftsman and process guru with a background in computer engineering. Rudra is the team’s formatting fanatic and APA whisperer, ensuring our final document looks sharp enough to cut diamonds.

**Kim:**

I am currently in the third semester of the Software Development program at the Southern Alberta Institute of Technology (SAIT), specializing in full-stack development. Their skills include proficiency in TypeScript, JavaScript, Python, Flask, React, Next.js, SQL Alchemy, SQLite, and Tailwind CSS. They have experience creating responsive user interfaces, developing efficient backend systems, and managing relational databases using modern frameworks and tools.

**Tehjib Singh:**    
 Perfectionist Innovator consistently delivers strategies that stand out and solve complex challenges. He is our quality control overlord, making sure every sentence sing and every report is on point. He possesses robust design skills and proficient command over design languages, enabling me to create visually compelling and technically sound solutions.

**Statement of Team Goals**    
 **We are not here to play—we are here to slay. Our goals:**

**·** Deliver a capstone project that screams "GOAT status" with a grade of 100% or higher (yes, we are gunning for extra credit).

·       Master software analysis tools while sharpening teamwork and problem-solving skills.

·       Build something, we can flex in our portfolios while learning to tackle team dynamics like pros.

**Assigned Roles**

**·       Team Leader:** Leona Motyer    
 Keeps the herd in formation, allocates tasks, and ensures the project’s success.

**·       Editor: Tehjib Singh**    
 The cleanup crew for typos, bad grammar, and formatting fails—polishing the project to perfection.

**·       Debugger: Kim**     
 Make sure our project meets client and assignment requirements.

**·       Deadline Tracker: Rudra**    
the clock-watching enforcer who ensures no one slacks off.

**Everyone contributes equally, and tasks are rotated to ensure balanced participation. If someone’s drowning, the team rallies—no GOAT gets left behind.**

**Established Norms and Expectations**

**·       Meeting Schedule:**    
 Core meetings are on Wednesdays and Fridays during class (online or in person). Flexible meetings are scheduled on Mondays, keeping everyone’s vibe in mind.

**·       Communication:**    
 Primary mode: Microsoft Teams. Backup: WhatsApp. Response times: Non-urgent (24 hours), urgent (by EOD—unless you are off saving the world).

**·       Collaboration Tools:**    
 Microsoft Teams and Google Docs because we are organized like that.

Conflict Resolution Plan

**·       First Offense:**    
 **Trigger:** Examples include missed meetings, incomplete or poor-quality work, or a hostile attitude without prior communication.    
 **Penalty:** Buy team snacks and drinks. A minimum of two members must approve the charge. Fuel the GOATS or feel the shame.

**·       Second Offense:**    
 **Trigger:** Repeated behavior or failure to improve after the first offense.    
 **Penalty:** A serious team discussion to recalibrate expectations. Additional work may be assigned. It requires 2-3 members to call it**.**

**·       Third Offense:**    
 **Trigger:** Persistent issues or blatant disregard for team rules and respect.    
 **Penalty:** Mark deduction on the assignment, up to a complete zero. A minimum of three members must approve. Severity decided case-by-case.

**·       Critical Fail:**    
 **Trigger:** Continued lack of accountability or sabotage.    
 **Penalty:** The team may vote to involve the instructor or remove offending members from the group. This is the nuclear option—nobody wants it, so keep it chill.

**Conclusion**    
 The GOATS are here to dominate this capstone project, combining brains, skills, and just the right amount of chaos to keep things interesting. Success is inevitable if we stick to the plan. Failure? Not on our watch.

**Signatures**

**·       Leona Motyer** Leona M

**·       Rudra Pratap**  *Rana jxtt*

**·       Yoohyun Kim**  YoohyunKim

**·       Tehjib Singh**  Tehjib Singh

**·       Chester Laraya**  C.Laraya

Workload Table Phase 1

|  |  |
| --- | --- |
| Leona Motyer | Use case diagram, descriptions, team contract and documentation |
| Tehjib Singh | Class diagram |
| Yoohyun Kim | PowerPoint and proposal compilation |
| Rudra Pratap | Class diagram |
| Chester Laraya | PowerPoint and formatting |

Workload Table Phase 2

|  |  |  |
| --- | --- | --- |
| **Task** | **Assigned To** | **Notes** |
| Preliminary UI Design | Chester | If made as a static web app, code can be reused |
| ERD Diagram | Kim |  |
| Use Case (Extended Format) & Documentation | Tehjib | Must be in APA format; refer to submission requirements. |
| Revised Class Diagram & Database Choice Explanation | Rudra | Includes input from Mamta. |
| Deployment Diagram | Leona |  |