# **CSC 431** TruckMuck System Architecture Specification (SAS)

**<Team number>**

|  |  |
| --- | --- |
| Carlos Carpio | Developer and QA Analyst |
| Ronald Vera | Project Lead & Designer |
| Patrick Johnston | Architect Lead & Developer |

# Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author(s) | Change Comments |
| 1.0 | 3/31/22 | Carlos Carpio, Ronald Vera, Patrick Johnston | First Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[1. System Analysis 6](#_Toc99646715)

[1.1 System Overview 6](#_Toc99646716)

[1.2 System Diagram 6](#_Toc99646717)

[1.3 Actor Identification 6](#_Toc99646718)

[1.4 Design Rationale 6](#_Toc99646719)

[1.4.1 Architectural Style 6](#_Toc99646720)

[1.4.2 Design Pattern(s) 7](#_Toc99646721)

[1.4.3 Framework 7](#_Toc99646722)

[2. Functional Design 8](#_Toc99646723)

[2.1 Vendor Diagram 8](#_Toc99646724)

[2.2 User Diagram 9](#_Toc99646725)

[3. Structural Design 10](#_Toc99646726)

# Table of Tables

System Analysis 1

System Overview 1.1

System Diagram 1.2

Actor Identification 1.3

Design Rationale 1.4

Architectural Style 1.4.1

Design Pattern(s) 1.4.2

Framework 1.4.3

Functional Design 2

Vendor Diagram 2.1

User Diagram 2.2

Structural Design 3

# Table of Figures

[System Diagram 6](#_Toc99646372)

[Vendor Diagram 8](#_Toc99646373)

[User Diagram 9](#_Toc99646374)

[Structural Design Diagram 10](#_Toc99646375)

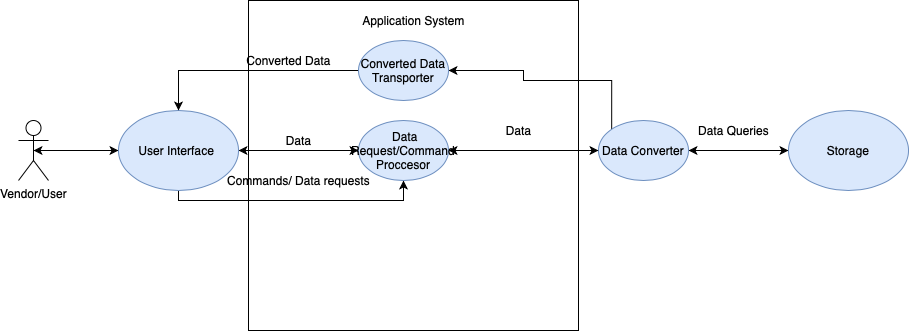
### System Analysis

#### System Overview

This document describes the architectural design and specifications for the TruckMuck mobile application. The system uses a three-tier architecture. The interface tier gives Users/vendors access to the application through an iOS compatible device. This then accesses the backend of the application, also known as the application layer. This includes all application logic and all resources needed to send data to the database. The database layer will hold important data and store efficiently.

The system is comprised of three main parts: The Data Request/command processor, the Data Converter, and the Converted Data transporter. The Data Request/Command Processor handles all data coming from the UI and sends it to the Data Converter once it is finished processing. The data converter stores this data into the database and sends any corresponding data to the Converted Data Transporter. This transporter will then decipher the data received and send it back into the UI for user interaction.

#### System Diagram



Figure

#### Actor Identification

There is two types of human actors: Users/customers, and Vendors. Users/customers will be able to see listings of trucks in their area and be able to order pickup ahead of time. Vendors will be allowed to upload a menu and location and will be able to accept orders.

#### Design Rationale

##### Architectural Style

Our system will be utilizing the 3-tier architectural style with a heavy emphasis on the back   
end components. This will be easy to do because we are going to utilize swifts ease of use UI creation to connect our Interface Layer and Application Layer seamlessly. Our three tiers we will be using is: Interface, Application, Database. The Interface Layer will deal strictly with the UI for both the vendor and the user while also sending data requests and command   
requests to the Application Layer. The Application Layer will deal with processing user data,   
taking command requests and data requests for things such as location tracking, finding   
menus, validation of creating accounts and logging in, social media, and connecting the   
database to the application and Interface Layer with the Application Layer as a buffer. The   
Database Layer will deal solely with archiving important data such as user logins, location   
tracking, and other things. The Database Layer will be connected to the Interface Layer and   
its data will be converted using the DB Interface Adapter.

##### Design Pattern(s)

Vapor is an amazing Swift Framework that will allow us to not run into many issues

pertaining to UI and Application layer due to both being in Swift. But with the Database

Layer being in MySQL, we will have to implement design patterns. We also will have to use

some in the Application Layer.

- Factory: We will use the factory method to allow objects with similar data classes,

such as Vendor and User Accounts and Menus, to be quickly made and distributed

while still maintaining uniqueness.

- Façade: We will use the façade method to break up the User and Vendor

accessibility in order to not simplify and consolidate code that both their data use

while still having restrictions for each of the respective User and Vendor.

- Adapter: We will use the adapter method to help convert data from database and

the application layers as they are in different languages. This will allow us to read,

input, and edit data much easier between the two layers.

- Command: We will use the command method to help add parameters that all

need to be met in order to use certain functions such as finding mutual food

trucks in close distance between User and Vendor, as you can’t just have ones

location and compile the data of mutual trucks between both of them.

- Stream: We will use the stream method to help streamline the process of using

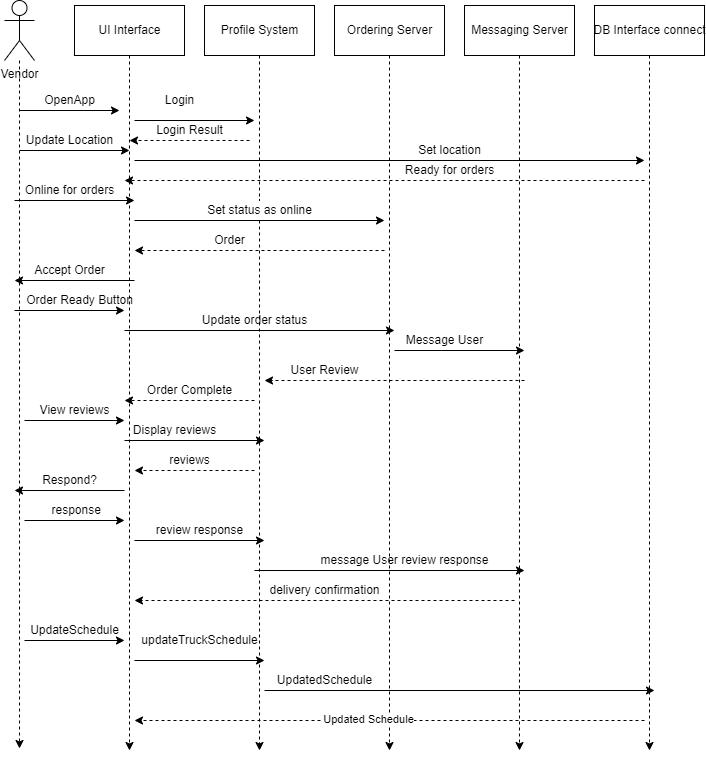
large amounts of flowing data and reading and writing it.

##### Framework

We will be using the Vapor Framework as it is compatible with Swift, our main coding   
language. Vapor will allow us to seamlessly take commands from the Interface and push   
them to the Application layer. Vapor also is very useful with MySQL as it has great   
compatibility, which will allow us to gather data from the Database and push it to the   
Application Layer. Vapor is great for the frontend as it has great UI functionality and is also   
great for the backend as Swift is its main language and it is extremely compatible and   
efficient with MySQL.

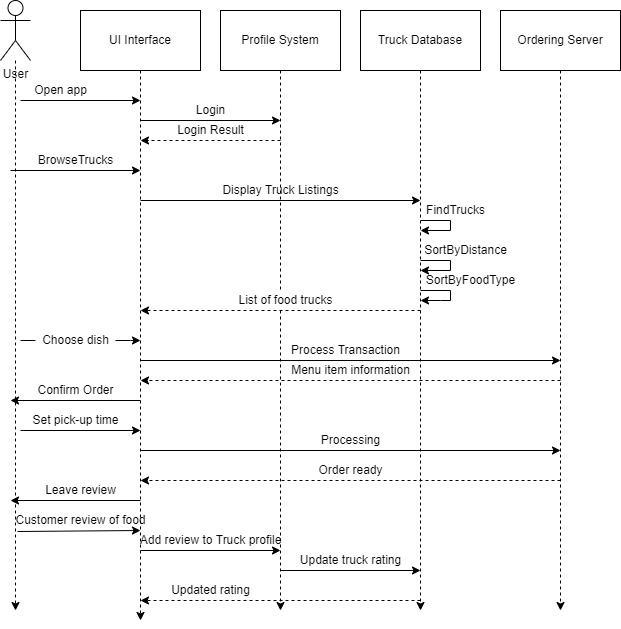
### Functional Design

#### Vendor Diagram



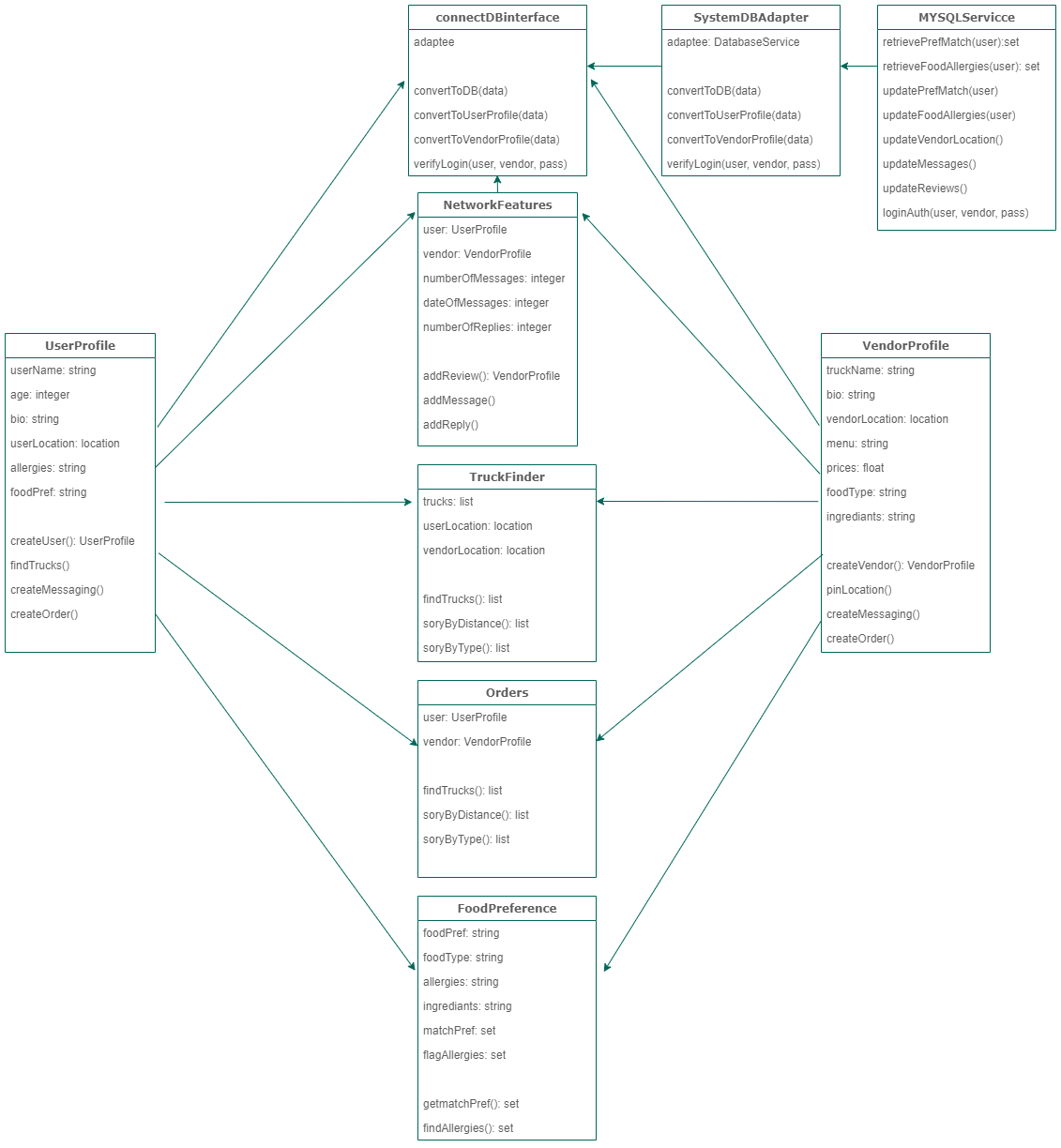
Figure

#### 2.2 User Diagram



Figure

### Structural Design



Figure