Architecture Description of

MVC for

SOEN solutions

Version 0.2

**Prepared By:**

Jake Lamothe - 40052279

[jlamothe75000@gmail.com](mailto:jlamothe75000@gmail.com)

Cristian Milatinov - 40098297

[milatinovcristian@gmail.com](mailto:milatinovcristian@gmail.com)

Kevin Ghannoum - 40088808

[kevin.ghannoum@gmail.com](mailto:kevin.ghannoum@gmail.com)

Contents

[**1 Introduction**](#_heading=h.o9yr0gs4d1xh) **2**

[**1.1 Identifying information**](#_heading=h.qk9tok644nfc) **2**

[**1.2 Supplementary information**](#_heading=h.bdj1u44zsxr0) **2**

[1.3 MVC architecture](#_heading=h.x4dj9pr0i830) 2

[**1.3.1 Model**](#_heading=h.vw3coqae4n0s) **3**

[**1.3.2 controller**](#_heading=h.6e1rz3apuj9k) **4**

[**1.3.3 view**](#_heading=h.qzz18ij43lgt) **5**

[**1.4 Other information**](#_heading=h.arqne5rm38wu) **7**

[**1.4.1 Architecture evaluations**](#_heading=h.wph9pl5hlf30) **7**

[**1.4.2 Rationale for key decisions**](#_heading=h.pdqc8ccgezl0) **7**

[**1.3.2.1 Node.js**](#_heading=h.lnxbz9) **7**

[**1.4.2.2 React.js**](#_heading=h.ci4ncxwfduoy) **7**

[**1.4.2.3 MySQL**](#_heading=h.ejxuv8p581y2) **7**

[**1.4.3 Acronyms and Abbreviations**](#_heading=h.2hq7uh8or8sg) **7**

[**2 Stakeholders and concerns**](#_heading=h.mhsz1n54oqjk) **9**

[**2.1 Stakeholders**](#_heading=h.bhhe3bh1147c) **9**

[**2.2 Concerns**](#_heading=h.rhl12on6goq1) **10**

[**2.2.1 Security**](#_heading=h.t0ro977l5py) **10**

[**2.2.2 Performance**](#_heading=h.k88x3oiyfze1) **10**

[**2.2.3 Reliability**](#_heading=h.jwa5tudy8v7n) **10**

[**2.2.4 Scalability**](#_heading=h.d3n2u8ed8b4y) **10**

[**2.3 Concern–Stakeholder Traceability**](#_heading=h.pxqj9e5fjnrl) **11**

[**3 Use Case view**](#_heading=h.ngprbchy6i4n) **12**

[**3.1 Logging In (issue # 3)**](#_heading=h.jmqabqamgzrp) **13**

[**3.2 Logging Out (issue # 7)**](#_heading=h.quyjog7w6uel) **15**

[**3.3 Requesting Transaction Log (issue # 17)**](#_heading=h.qs7zzbe55knj) **17**

[**3.4 Placing an Order (issue # 16)**](#_heading=h.wpo3nvh0tpy2) **19**

[**3.5 Creating, Removing and Editing Items (issue # 10)**](#_heading=h.adwms3l6wqw0) **22**

[**3.6 Shipment (issue # 24)**](#_heading=h.7ybn5mvjepjf) **25**

[**3.7 Managing Shipments and Inventory (issue # 24,20)**](#_heading=h.8l251n1i49bf) **28**

[**3.8 Creating, Removing and Editing Locations (issue # 18)**](#_heading=h.ym7xcdddinx5) **31**

[**3.9 Recovering or Changing a Password (issue # 5)**](#_heading=h.cej3dbhiouq9) **34**

[**3.10 Registering a Business Account (issue # 2)**](#_heading=h.jt7605s2l5si) **36**

[**4 Logical view**](#_heading=h.yk5zidj0388m) **38**

[**5 Physical view**](#_heading=h.x5z486utubo8) **39**

[**6 Inconsistencies**](#_heading=h.8nv3qgaq8vlg) **40**

[**7 Bibliography**](#_heading=h.mfzahsib4o) **40**

# 1 Introduction

## 1.1 Identifying information

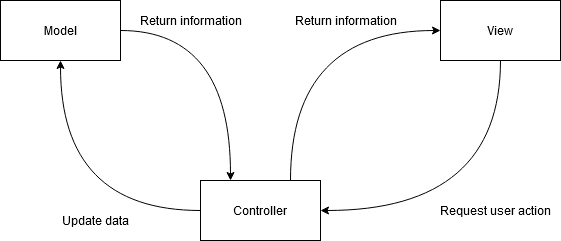
This document describes the MVC architectural system that is to be used to achieve our ERP system. The ERP system is meant to allow a business to manage their resources (time, money, raw goods, final products, etc) in one service. This will also allow the company to reduce the complexity of managing their resources as well as making it faster to gather a large amount of data at once (receipts, inventory, etc).

## 1.2 Supplementary information

Version 0.2 of the architectural document, made as of 2021-02-23. This document covers the initial design plans and scope up to our 5th and final deliverable. The development style that the team will be using is predominately scrum with a mix of pair programming. the pair programming will be used on occasion if a team member needs help understanding a new section of code or wants confirmation that they are using the methods correctly. The planned launch date will be around april 20th with the possibility of that being pushed back if more polish is needed.

## 1.3 MVC architecture

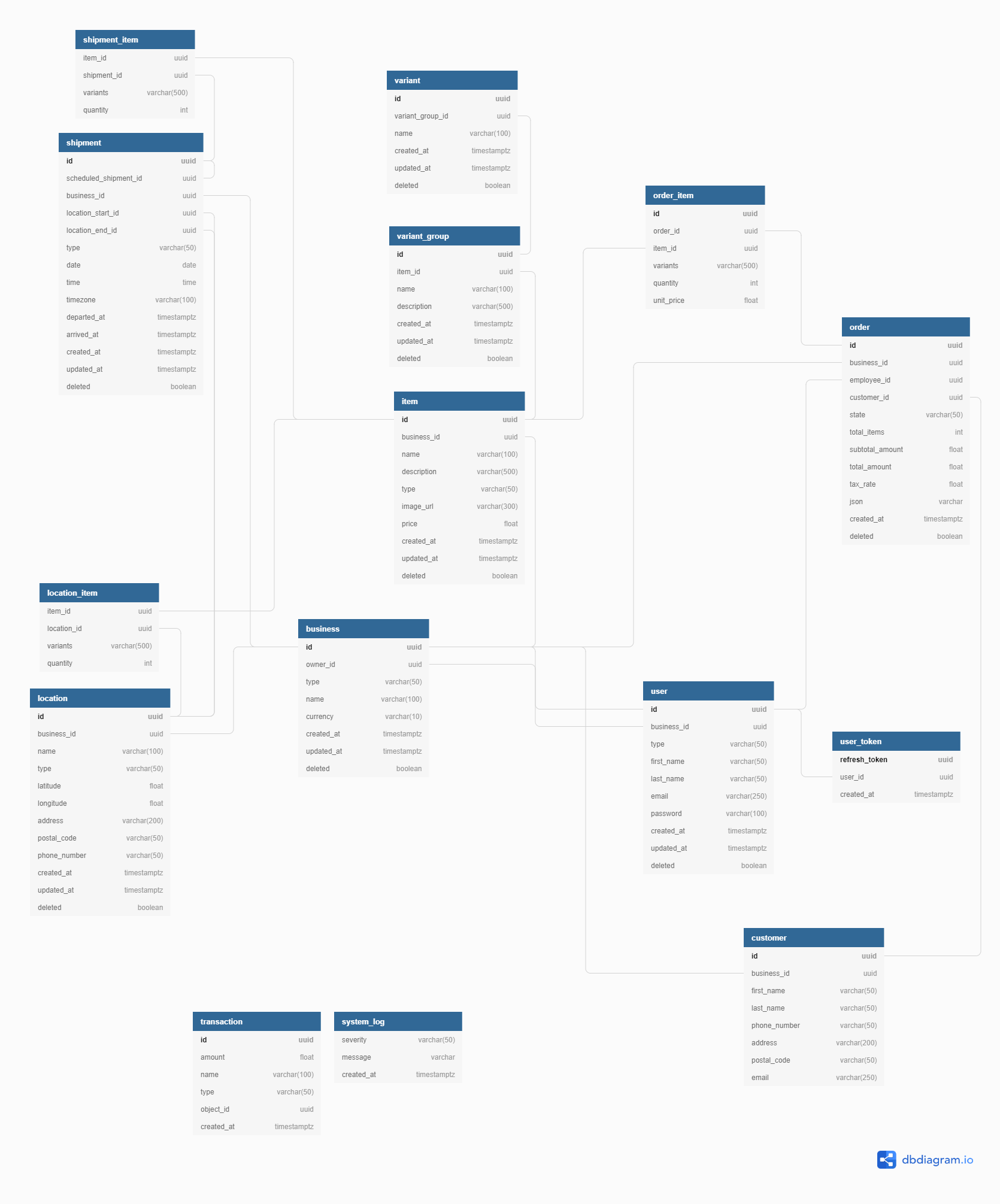
The system will be using an mvc design pattern in order to keep the design of the architecture simple. This is done simply enough by having a designated frontend, backend and database. The frontend (react) acts as the view of the mvc and is updated by the controller and is also where users perform actions that the controller uses to update the view and the model. the backend(node js) acts as the controller of the mvc and updates the view and model while accepting requests from the view and receiving information from the model when prompted. Our database. Lastly, the backend (mysql) acts as the model. The model's responsibility is to store information, and update this information when prompted by the controller as well as returning information to the controller when necessary.



**figure 1:** *MVC diagram*

### 1.3.1 Model

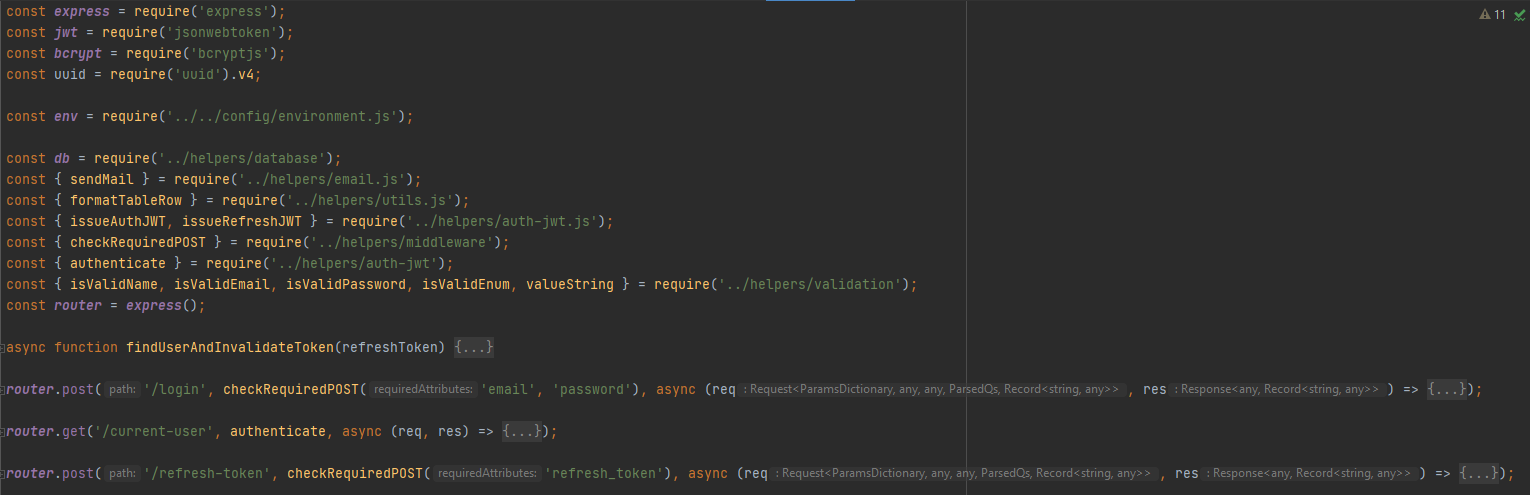
figure 2 depicts the “model” of the mvc system. it is responsible for updating the data based on requests from the controller as well as returning information (from queries).a logical view is provided in section 4.



**figure 2:** *Database structure (model)*

### 1.3.2 controller

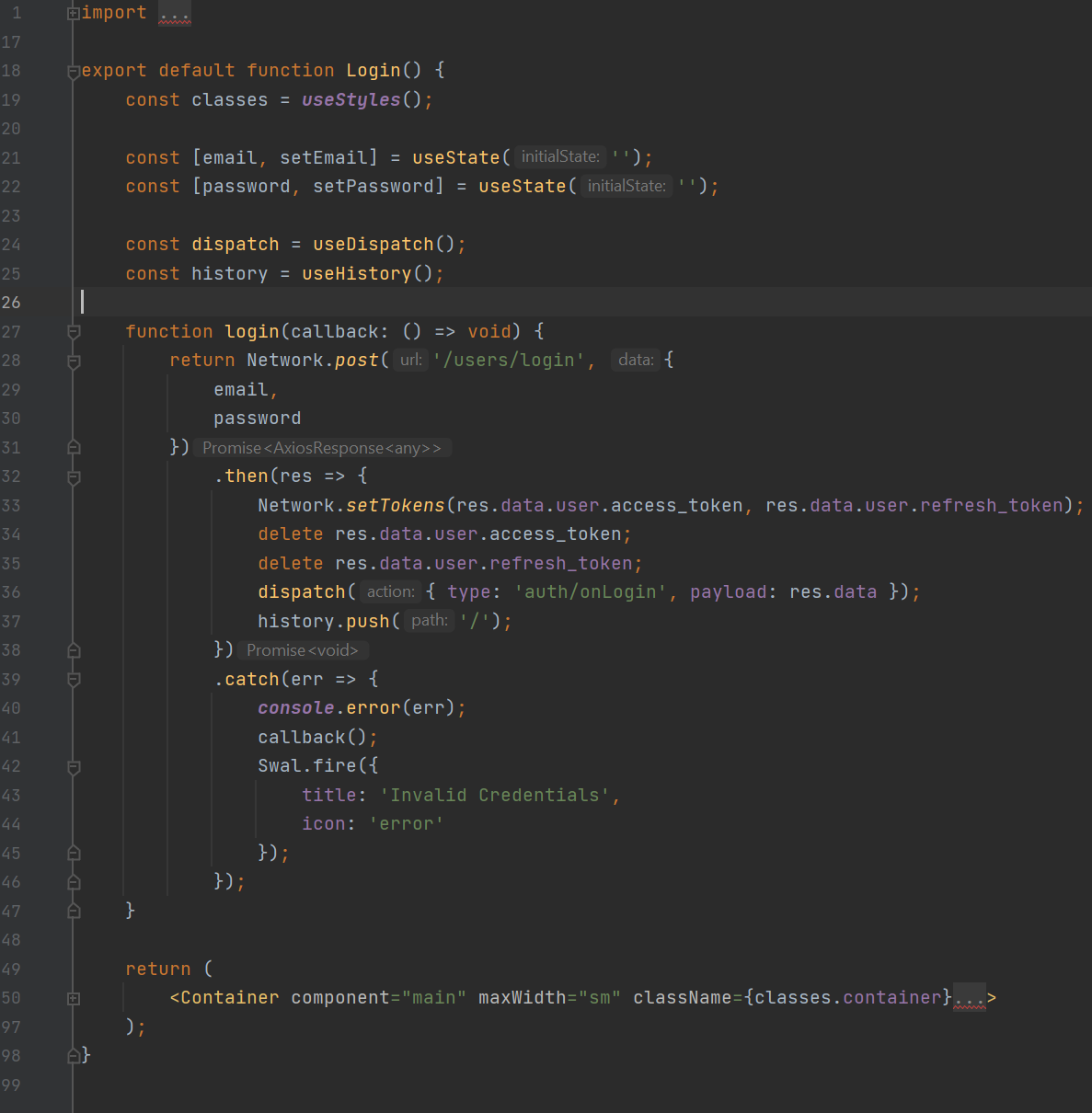
Figure 3 depicts the controller of the mvc architecture. This is where all the requests from the front end are done and processed, and the result gets returned to the front end. Additionally it sends requests to the model to update data and can query it for information. A physical view can be seen in section 5 that shows how the backend is broken up into controllers.



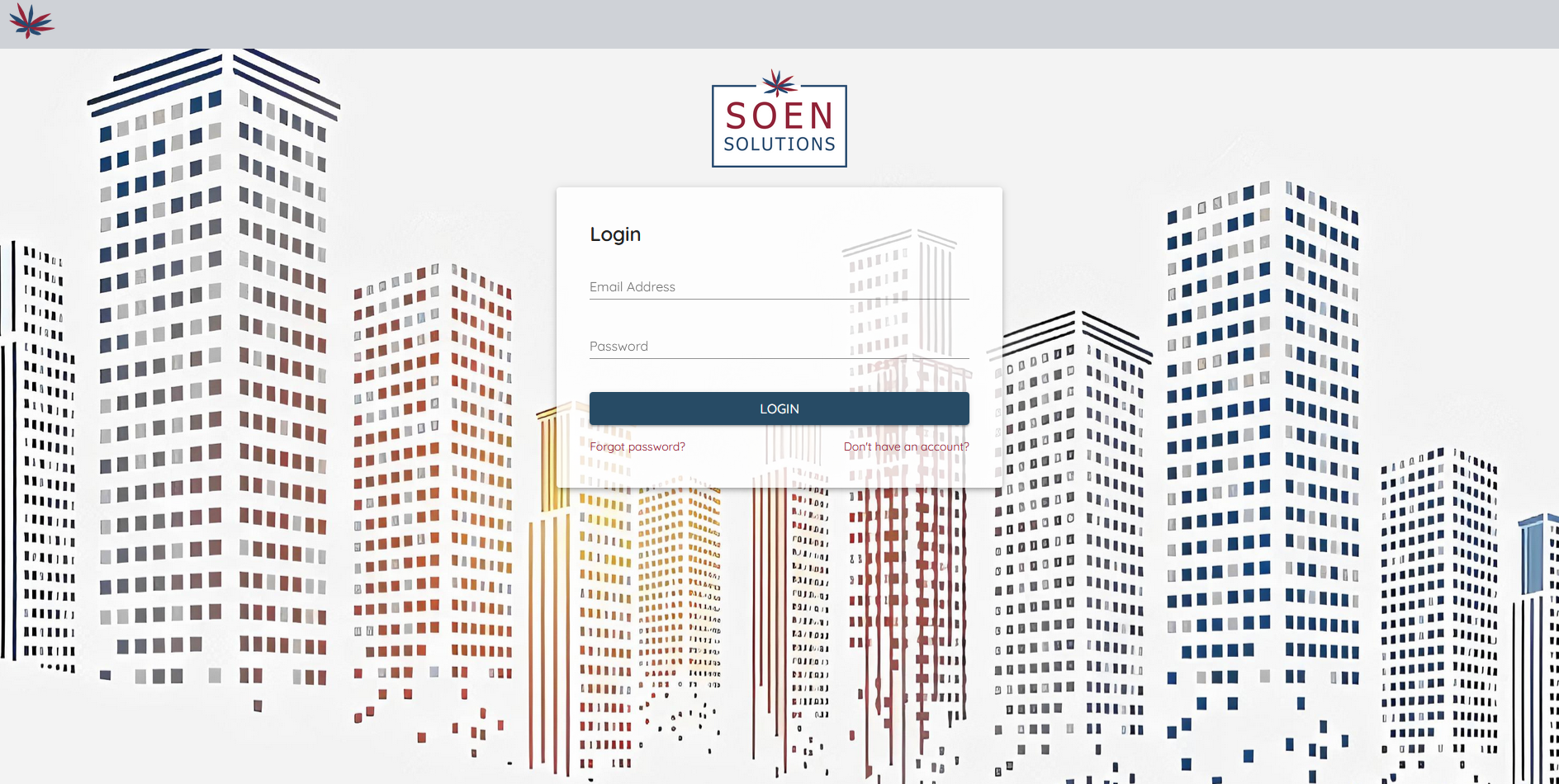
**figure 3:** *Controller code (backend)*

### 1.3.3 view

These two figures depict the view (frontend) of the mvc architecture. its responsibilities is to provide the user with an interface, send user requests to the controllers and update based on information received form the controllers. An example of the user interactions that cause actions can be found in section 3 (although this is not restricted to the view, the view is represented by the information being sent to and from the user actor).



**figure 4:** *View code (front end)*

**figure 5:** *View interface (backend)*

## 1.4 Other information

This section lists additional information regarding the architecture of the system.

### 1.4.1 Architecture evaluations

We are using MVC to keep our design simple and easy to pipeline. This architecture was chosen due to it being quick to code and easy to test (as one of our criteria was testability and a specific code coverage). We are aware that this architecture does have some problems such as the tendency to make tight couplings which result in code being dependent on each other and harder to update and change.

### 1.4.2 Rationale for key decisions

#### 1.3.2.1 Node.js

We chose node.js for the backend due to high familiarity and because it is a well suited framework for server side applications

#### 1.4.2.2 React.js

We chose React for the front end framework due to high familiarity and due to its components feature.

#### 1.4.2.3 MySQL

We chose MySQL for the back end (tentatively) due to high familiarity. we may switch to MongoDB if mysql proves to be insufficient for our requirements.

### 1.4.3 Acronyms and Abbreviations

**ERP** - Enterprise Resource Planning. ERP is software that is used to help businesses plan and track their resources

**SQL** - Structured Query Language. SQL is the most common language used to communicate with database servers.

**MySQL** - Implementation of SQL designed and created by the software company Oracle (<https://www.oracle.com/index.html>).

**MVC** - Model-View-Controller. MVC is an architectural design pattern most often used in web applications. It allows the separation of the interface (view) from the logic (controller) it executes and data structures (model) it manipulates.

**CEO** - Chief Executive Office. The CEO of a company is in charge of its operations and important decisions.

**UML** - Universal Modelling Language. UML is a set of visual standards used to express various software related concepts. UML is used in the rest of this document.

**HTTP** - Hypertext Transfer Protocol. The protocol used to deliver web pages to web browsers. HTTP can also be used to transfer data between a client and a server.

**API** - application programming interface. API’s are used to allow different types of software to interact with each other.

# 

# 2 Stakeholders and concerns

## 2.1 Stakeholders

The following table lists the stakeholders involved in the development, use and maintaining of the system-of-interest.

|  |  |
| --- | --- |
| **Name** | **Description** |
| CEO’s | The person or entity who makes the most important decisions. |
| Administrators | The people who will be in charge of maintenance and have the most control of the system |
| Employees | The people employed by the company for various tasks such as production or sales. |
| Warehouse Managers | The people who manager the individual warehouses |
| Customers | The people who are buying the products. |
| Company Owner | The person or entity who owns the company. |
| Suppliers | The persons responsible for delivering the materials to the company. |

**Table 6:** *List of stakeholders concerned with the system, along with their descriptions.*

## 

## 2.2 Concerns

### 2.2.1 Security

The system must not store passwords in a plain text format. The system must encrypt and restrict the flow of sensitive information. Sensitive information includes customer contact information, location addresses, employee details and potentially more. This is arguably the most important concern which will affect the most stakeholders.

### 2.2.2 Performance

The system must not exceed load times of over 5 seconds and the user interface should be responsive and not impede the users. When implemented, the system should aim to reduce required memory to a minimum.

### 2.2.3 Reliability

The system is expected to be operational for long periods of time. Any maintenance done to the system must be announced with reasonable foresight and scheduled during low usage timeslots. Features must be tested extensively to prevent and reduce failures at runtime. Whenever possible, testing should be automated to save time.

### 2.2.4 Scalability

The system must support a high number of users operating various tasks and pages at the same time. A preliminary estimation of the amount of users operating the application at the same time might range between 100-500 concurrent users. The architecture of the system must allow for more processing power to easily be added to the existing system as this kind of system can rapidly grow in terms of users.

## 2.3 Concern–Stakeholder Traceability

The following table maps the concerns listed in section 2.2 to the appropriate stakeholders.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | CEOs | Administrators | Employees | Warehouse Managers | Customers | Company owner | Suppliers |
| Security | x | x | - | - | x | x | x |
| Performance | x | - | x | x | x | x | - |
| Reliability | x | - | x | x | x | x | - |
| Scalability | x | x | - | - | - | x | - |

**Table 7:** Concern-to-Stakeholder Associations

The CEOs and company owner(s) are concerned with all aspects of the system as any failure will affect their bottom line. they need performance to not slow down the flow of business, if the system is unreliable they will lose money in down time. Security matters as a breach in security could result in confidential information of the company being stolen or customers not wanting to do business. And scalability matters as they need to know if they can keep this system if they plan to expand or not.

Suppliers are only concerned with the security of the system as they do not interact with the system directly but would like to know that their information is secure.

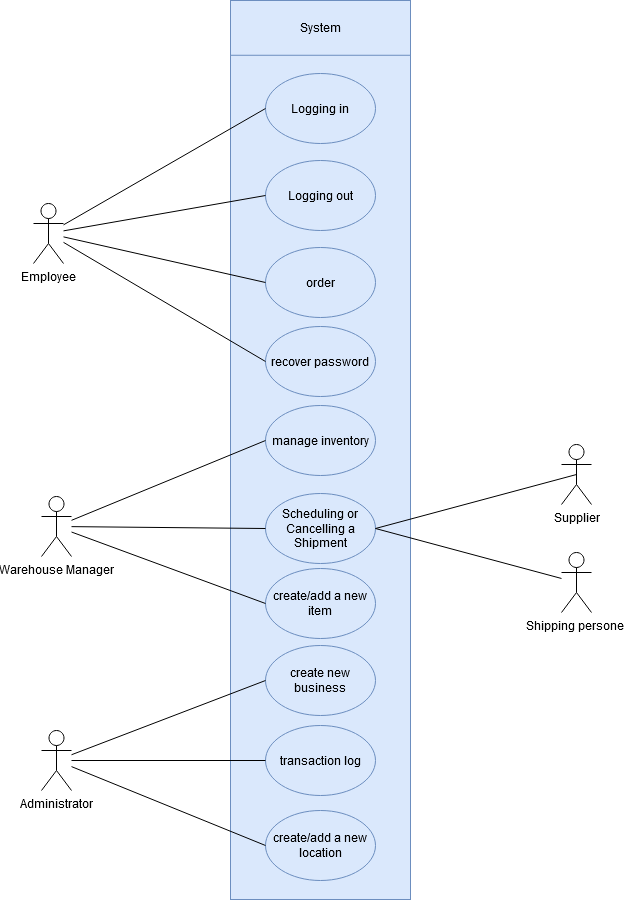
The customers are concerned with the security, performance and reliability of the system. they are concerned with security as their information will be stored into this service and they would expect that information to be secure. additionally they are also concerned with performance and reliability as if either is particularly lacking then it could increase their wait time and reduce customer satisfaction.

Administrators are concerned with security and scalability as they need to know for what size of company the system is designed for and whether or not it can be scaled to a larger one. they are also concerned with security as they will be involved in the process of ensuring the security of the information inputted into the system withing their capabilities. (e.g. managing roles )

employees and warehouse managers are concerned with the performance and reliability of the system as they will be using the system the most and be the most affected by unnecessarily slow performance and long down times.

# 3 Use Case view

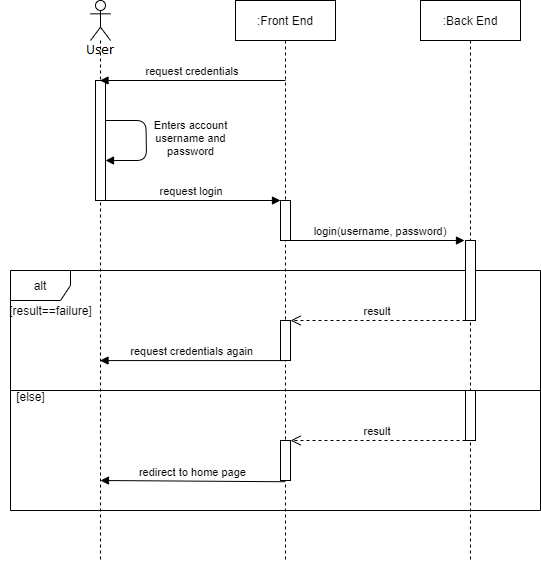
The purpose of the use case view is to identify the architectural elements and how the system will communicate and behave. This section contains the use cases and their description and related UML diagrams (use case diagrams and sequence diagrams) to illustrate how they communicate with the system. Use case 3.1 to 3.10 are from different milestones that were chosen as they showcase the most different aspects of our system with the least amount of use cases to succinctly demonstrate the system.



**Figure 8:** The *use case diagram for the system.*

## 3.1 Logging In (issue # 3)

|  |  |
| --- | --- |
| **Use case name** | Logging in |
| **Level** | User-level |
| **Brief description** | Users will be able to login into their account. Each account has a different access level depending on the person using it. Access levels listed in order of highest to lowest privileges are Owner, Administrator, Manager, Employee. When an access level is required, access levels with higher privilege than the one mentioned are assumed to be able to execute the operation. |
| **Preconditions** | * The user is not already logged in * The user must have an existing account |
| **Triggering event** | The user submits his credentential on the login page |
| **Main flow** | 1. The user is redirected to the login page upon navigating to the website. 2. The system prompts the user for their credentials. 3. The user inputs their credentials. 4. The system authenticates the user with a token and registers the successful login attempt. |
| **Extensions** | 3a) Invalid credentials   1. If the user enters invalid credentials, the system informs the user of the error and rejects the login attempt. The use case continues from step 2. |
| **Postconditions** | * The user is redirected to the home page * The user is now authenticated and is given access to the system’s functionalities |

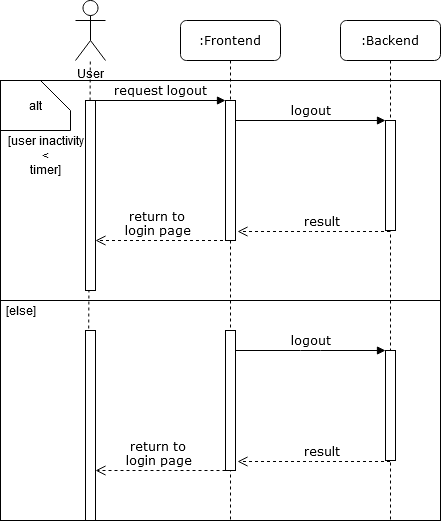


**Figure 8.1:** *Sequence diagram presenting the login flow of the system.*

## 

## 3.2 Logging Out (issue # 7)

|  |  |
| --- | --- |
| **Use case name** | Logging out |
| **Level** | User-level |
| **Brief description** | Users will be able to logout of their accounts. Once logged out, a user will not be able to access application features and must login again in order to continue to use the system. |
| **Preconditions** | * User must be already logged into their account |
| **Triggering event** | User request to logout from his account |
| **Main flow** | 1. The user requests to log out of their account. 2. The system invalidates the user’s authentication token and removes the user’s access to application features. 3. The system redirects the user to the login page. |
| **Extensions** | 1a) User inactive   1. If the user has been inactive for a certain amount of time, the system automatically logs out the user. |
| **Postconditions** | * The user has been logged out of their account * The user has been redirected to the login page |



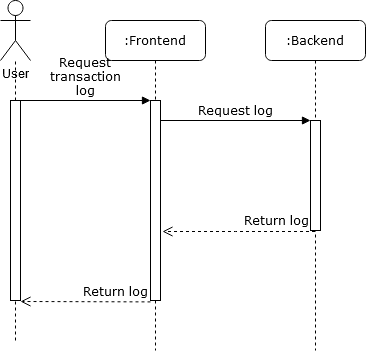
**Figure 8.2:** *Sequence diagram for logging out use case*

## 

## 

## 3.3 Requesting Transaction Log (issue # 17)

|  |  |
| --- | --- |
| **Use case name** | Requesting transaction log |
| **Level** | User-level |
| **Brief description** | Administrator may consult logs of transactions processed by the system. A transaction corresponds to a money transfer between company members or customers (ex: purchase, employee salary). |
| **Preconditions** | * The user must be logged in * The user must be an administrator |
| **Triggering event** | The user requests the transaction log |
| **Main flow** | 1. The user requests the transaction log from the system with specific search criteria (ex: amount, time frame). 2. The system fetches the requested information from the database. 3. The system displays the list of transactions corresponding to the user’s search criteria. |
| **Extensions** | N/A |
| **Postconditions** | N/A |

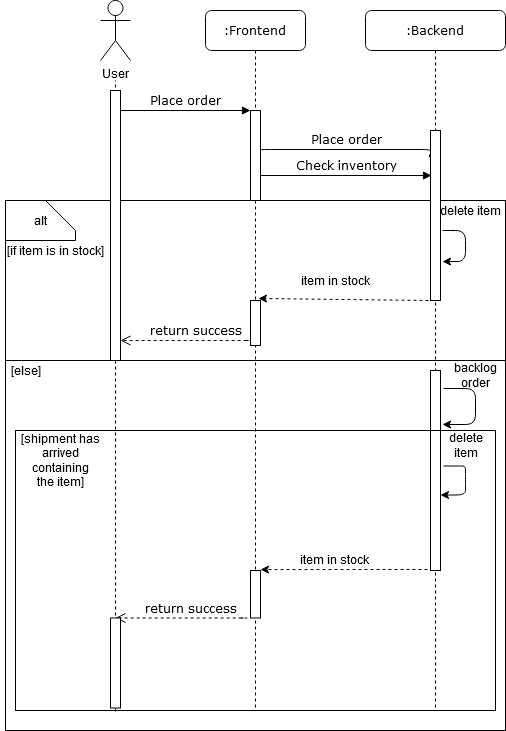


**Figure 8.3:** *Sequence diagram for requesting transaction log use case*

## 

## 3.4 Placing an Order (issue # 16)

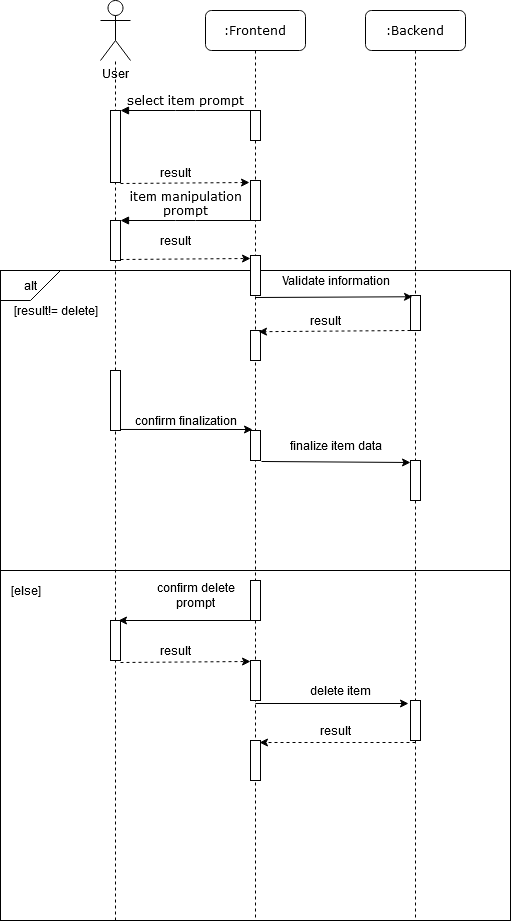
|  |  |
| --- | --- |
| **Use case name** | Placing an order |
| **Level** | User-level |
| **Brief description** | Employees may place orders in the system on behalf of customers of the business. When placing an order, the required information such as purchased items is filled in by the employee and the system attempts to fill the order if stock is available. |
| **Preconditions** | * The user must be logged in * The user must be an employee |
| **Triggering event** | The user chooses to create a new order. |
| **Main flow** | 1. The system prompts the user to enter the required information for the order. 2. The user inputs the requested information into the system  (ex: cost, items, customer). 3. The system verifies that the required items are in stock. 4. The system removes the necessary items from the inventory. 5. The system fills the order and processes the transaction associated with it. |
| **Extensions** | 3a) Item not in stock   1. The system considers the order as “backlogged”. 2. When new shipments arrive, the system will check to see if any orders can be filled with certain items. 3. If some orders are newly completed, the system will automatically process the needed transactions. The use case ends successfully. |
| **Postconditions** | * The order is filled * The inventory of the location at which the order was created is properly updated |



**Figure 8.4:** *Sequence diagram describing the ordering flow.*

## 3.5 Creating, Removing and Editing Items (issue # 10)

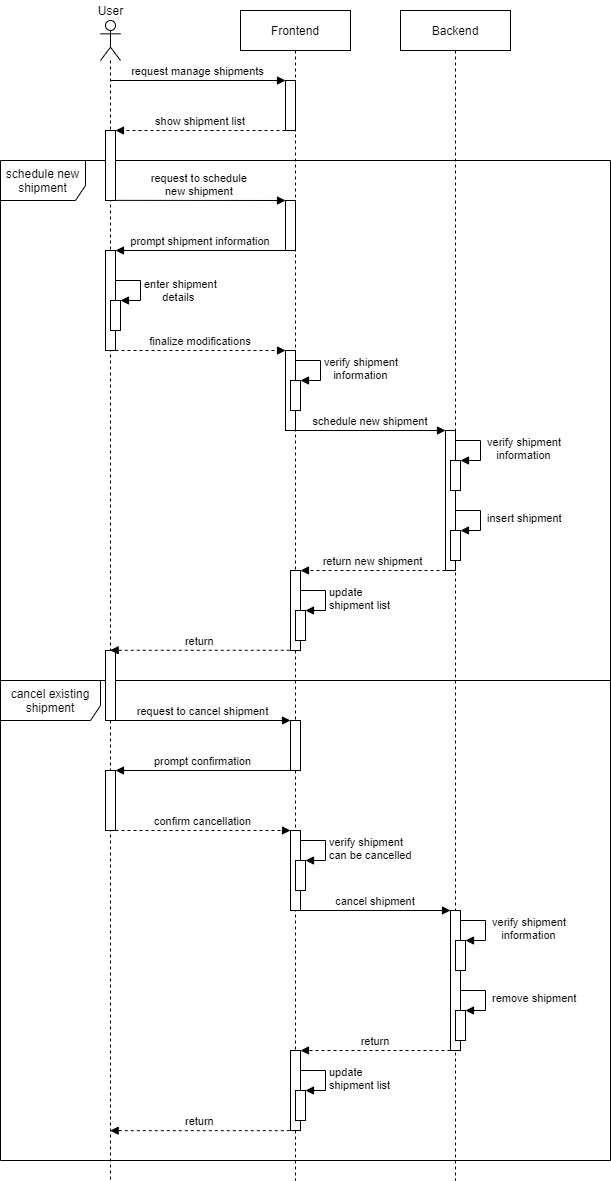
|  |  |
| --- | --- |
| **Use case name** | Creating, removing and editing items |
| **Level** | User-Level |
| **Brief description** | When setting up a new business, administrators will be required to fill out a list of items that are part of the production and sales chains of their business. Administrators can create new items, change the information of existing items or remove unwanted items. |
| **Preconditions** | * User must be authenticated * User must be an administrator |
| **Triggering event** | The user selects the option to modify their business’ item list. |
| **Main flow** | 1. The system displays the list of all item types currently created in the business. 2. The user selects the item they would like to modify or chooses to create a new item. 3. The system prompts the user to enter or modify any information needed pertaining to the item. 4. The user enters the information asked from the system. 5. The system validates the information according to specific rules (ex: item name should be at most 50 characters). 6. The user chooses to finalize their modifications. 7. The system executes the operation and updates the item information accordingly in the database. |
| **Extensions** | 2a) Creating a new item   1. If the user selects the option to add a new item, they will not be given the option to delete it in step 4. Instead, they will have the option to cancel the operation. The use case continues as normal from step 3.   4a) Deleting an item   1. If the user selects the option to delete the selected item, the system prompts the user to confirm the action, informing them that it is a destructive and dangerous operation. 2. The user confirms or denies the operation. 3. If the user confirms the action, the system executes the operation, and thereby deletes the item from the database and from any location inventories. The use case ends successfully. 4. Otherwise, if the user denies the action, the use case continues as normal from step 4.   4b) Cancelling the creation/modification of an item   1. If the user chooses to cancel the addition or modification of an item, the system prompts the user for a confirmation, informing them that they would lose any data entered in the form. 2. The user confirms or denies the operation. 3. If the user confirms the action, the system closes the dialog shown in step 4 and the use case ends unsuccessfully. 4. Otherwise, if the user denies the action, the use case continues as normal from step 4.   5a) Invalid item information   1. If the system detects invalid fields, it informs the user of the invalid fields and of the reason why. 2. The system prevents the user from continuing until all entered fields are valid. The use case continues as normal from step 4. |
| **Postconditions** | * An operation was successfully performed on an item (create, edit, delete) * The changes from the operation are reflected in the database |



**Figure 8.5:** *Sequence diagram for creating, removing and editing items use case*

## 3.6 Shipment (issue # 24)

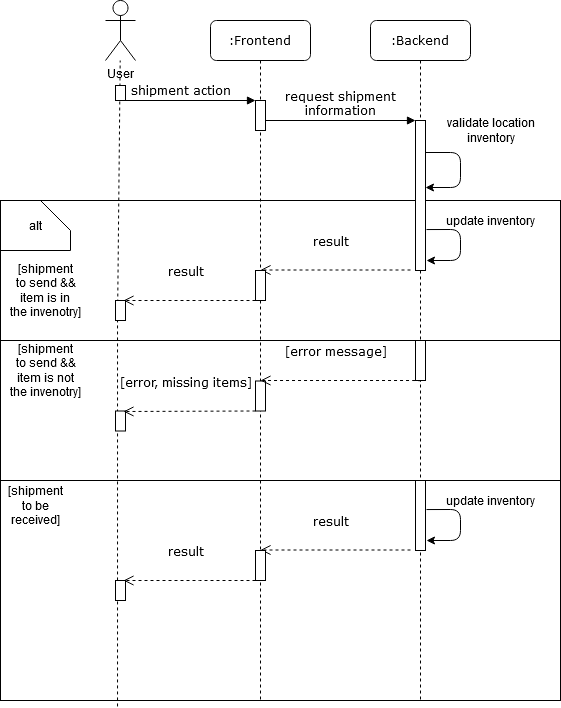
|  |  |
| --- | --- |
| **Use case name** | Shipment |
| **Level** | User-level |
| **Brief description** | Shipments are defined as the transportation of items between internal business locations (ex: from warehouse to store, from supplier to warehouse). Warehouse managers are given the ability to schedule shipments of various items at specific dates to execute internal inventory transfers between locations. |
| **Preconditions** | * User must be authenticated * User must be an manager |
| **Triggering event** | The user selects the option to manage their business’ shipments. |
| **Main flow** | 1. The system displays the list of currently scheduled shipments. 2. The user chooses to cancel an existing shipment or schedule a new shipment. The user cannot choose to cancel a shipment that is already in progress. 3. The system prompts the user for any information required to execute the operation. 4. The user enters the information requested by the system. 5. The system validates the information according to specific rules (ex: shipment must have a start and end location). 6. The user chooses to finalize their modifications. 7. The system executes the operation and updates the shipment information accordingly in the database. |
| **Extensions** | 2a) Cancelling an existing shipment   1. If the user selects the option to cancel a specific shipment, the system prompts the user to confirm the action, informing them that it is a destructive and dangerous operation. 2. The user confirms or denies the operation. 3. If the user confirms the action, the system executes the operation, and thereby removes the shipment from the database. The use case ends successfully. 4. Otherwise, if the user denies the action, the use case continues as normal from step 1.   5a) Invalid shipment information   1. If the system detects invalid fields, it informs the user of the invalid fields and of the reason why. 2. The system prevents the user from continuing until all entered fields are valid. The use case continues as normal from step 4. |
| **Postconditions** | * An operation was successfully performed on a shipment (create, cancel) * The changes from the operation are reflected in the database |



**Figure 8.6:** *Sequence diagram for scheduling or cancelling a shipment use case*

## 3.7 Managing Shipments and Inventory (issue # 24,20)

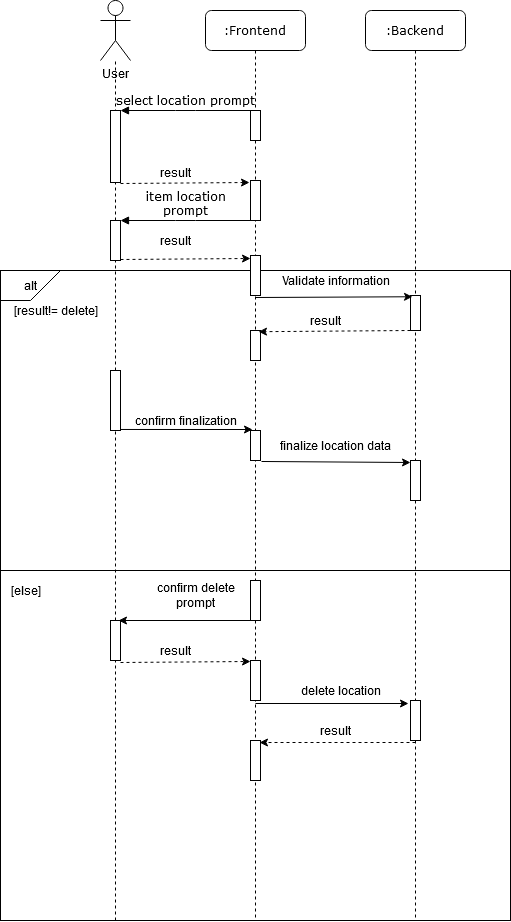
|  |  |
| --- | --- |
| **Use case name** | Managing shipments and inventory |
| **Level** | User-level |
| **Brief description** | Employees can specify when a shipment arrives or departs from a location. The location’s inventory is automatically updated by the system to reflect the item quantities after the shipment’s execution. |
| **Preconditions** | * The user must be authenticated * The user must be an employee |
| **Triggering event** | The user specifies that a shipment is arriving or leaving a location. |
| **Main flow** | 1. The system retrieves the information for the specified shipment. 2. The system validates that the location must have the required inventory to execute the transfer. 3. The system updates any changes to the inventory in the database. |
| **Extensions** | 2a) Shipment to send   1. If the location’s inventory does not contain the items necessary, the system informs the user that the shipment is impossible and the use case fails. 2. Otherwise, the system removes the items of the shipment from the location’s inventory and the use case continues from step 3.   2b) Shipment received   1. The system adds the items of the shipment to the location’s inventory. The use case continues from step 3. |
| **Postconditions** | * The shipment has been confirmed as departed or arrived * The inventory has been updated and quantities have been modified |

**Figure 8.7:** *Sequence diagram for managing shipments and inventory use case*

## 

## 3.8 Creating, Removing and Editing Locations (issue # 18)

|  |  |
| --- | --- |
| **Use case name** | Creating, removing and editing locations |
| **Level** | User-Level |
| **Brief description** | When setting up a new business, administrators will be given the option to add locations pertaining to their business. A business must have at least one location at all times. |
| **Preconditions** | * User must be authenticated * User must be an administrator/owner |
| **Triggering event** | The user selects the option to modify their business’ location list. |
| **Main flow** | 1. The system displays the list of all locations currently in the business. 2. The user selects the location they would like to modify or chooses to create a new location . 3. The system prompts the user to enter or modify any information needed pertaining to the location. 4. The user enters the information asked from the system. 5. The system validates the information according to specific rules (ex: item name should be at most 50 characters). 6. The user chooses to finalize their modifications. 7. The system executes the operation and updates the location information accordingly in the database. |
| **Extensions** | 2a) Creating a new location   1. If the user selects the option to add a new location, they will not be given the option to delete it in step 4. Instead, they will have the option to cancel the operation. The use case continues as normal from step 3.   4a) Deleting a location   1. If the user selects the option to delete the selected location, the system prompts the user to confirm the action, informing them that it is a destructive and dangerous operation. 2. The user confirms or denies the operation. 3. If the user confirms the action, the system executes the operation, and thereby deletes the location from the database. The use case ends successfully. 4. Otherwise, if the user denies the action, the use case continues as normal from step 4.   4b) Cancelling the creation/modification of a location   1. If the user chooses to cancel the addition or modification of a location, the system prompts the user for a confirmation, informing them that they would lose any data entered in the form. 2. The user confirms or denies the operation. 3. If the user confirms the action, the system closes the dialog shown in step 4 and the use case ends unsuccessfully. 4. Otherwise, if the user denies the action, the use case continues as normal from step 4.   5a) Invalid location information   1. If the system detects invalid fields, it informs the user of the invalid fields and of the reason why. 2. The system prevents the user from continuing until all entered fields are valid. The use case continues as normal from step 4. |
| **Postconditions** | * An operation was successfully performed on a location (create, edit, delete) * The changes from the operation are reflected in the database |



**Figure 8.8:** *Sequence diagram for creating, removing and editing locations use case*

## 3.9 Recovering or Changing a Password (issue # 5)

|  |  |
| --- | --- |
| **Use case name** | Recovering or changing a password |
| **Level** | User-Level |
| **Brief description** | Users may forget their passwords or may simply want to change them. In such cases, the system can send a link to a user’s email which they may use to reset their password. |
| **Preconditions** | * The user must have a registered account |
| **Triggering event** | The user selects the password recovery option. |
| **Main flow** | 1. The system sends an email with a link allowing the user to reset their password. 2. The user follows the given link to the reset form. 3. The system prompts the user for their new password. 4. The user enters their new password. 5. The system validates the password according to a predefined set of rules (ex: password length > 8, etc..). 6. The system changes the user account’s password. |
| **Extensions** | 5a) Invalid password   1. If the user enters an invalid password, the system displays an error message an informs the user of the unsatisfied criteria. The use case continues from step 4. |
| **Postconditions** | * The user’s password was changed |

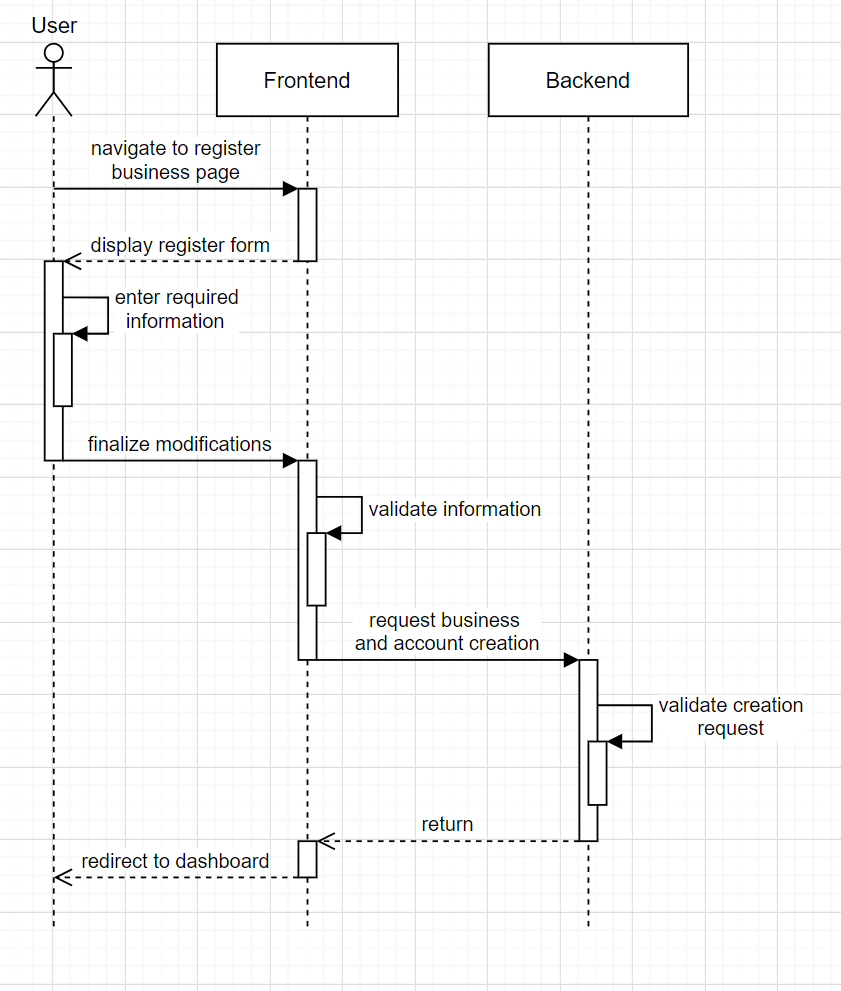
## 

## 

**Figure 8.9:** *Sequence diagram for Recover Password*

## 3.10 Registering a Business Account (issue # 2)

|  |  |
| --- | --- |
| **Use case name** | Registering a business account |
| **Level** | User-Level |
| **Brief description** | When a business wants to join the platform and have access to its features, the business owner must create an account on the platform. They will need to provide information relating to their business (ex: name, phone number, business type) as well as a valid email address. Both the business and the owner account will be created as a result. |
| **Preconditions** | N/A |
| **Triggering event** | The user selects the option to register their business on the platform. |
| **Main flow** | 1. The system prompts the user for the information needed to register a business. 2. The user inputs the information required by the system. 3. The system validates the information given by the user, checking if a business with the given name already exists or if an account with the given email already exists. 4. The system creates the new business and inserts it into the database. 5. The system creates the owner account using the given credentials. |
| **Extension s** | 3a) Invalid business or account details   1. If the user inputs invalid information, the system displays an error message informing the user of the incorrect fields. The use case continues from step 3. |
| **Postconditions** | * The business is registered in the system * The business’ owner account has been created, and can now be used to login |

**

**Figure 8.10:** *Sequence diagram for registering a business.*

# 4 Logical view

The purpose of the logical view is to describe the various data structures that will be present in the system and how they are related to one another. It is concerned with the functionality of the system as a whole and often expressed through domain model diagrams.

The domain model diagram for the system-of-interest is shown in figure 4. The diagram expresses the different types of entities in the system and the relationships between them. The database will be closely modeled to this diagram with slight changes to accommodate the SQL practice. This diagram was created to allow the system to work with many different types of items and end-products without over complicating the design, or losing performance in lookup costs. Each entity shown in this diagram will be converted into a table of the database. An extra logging table is planned in which various errors or significant events will be recorded in brief detail.

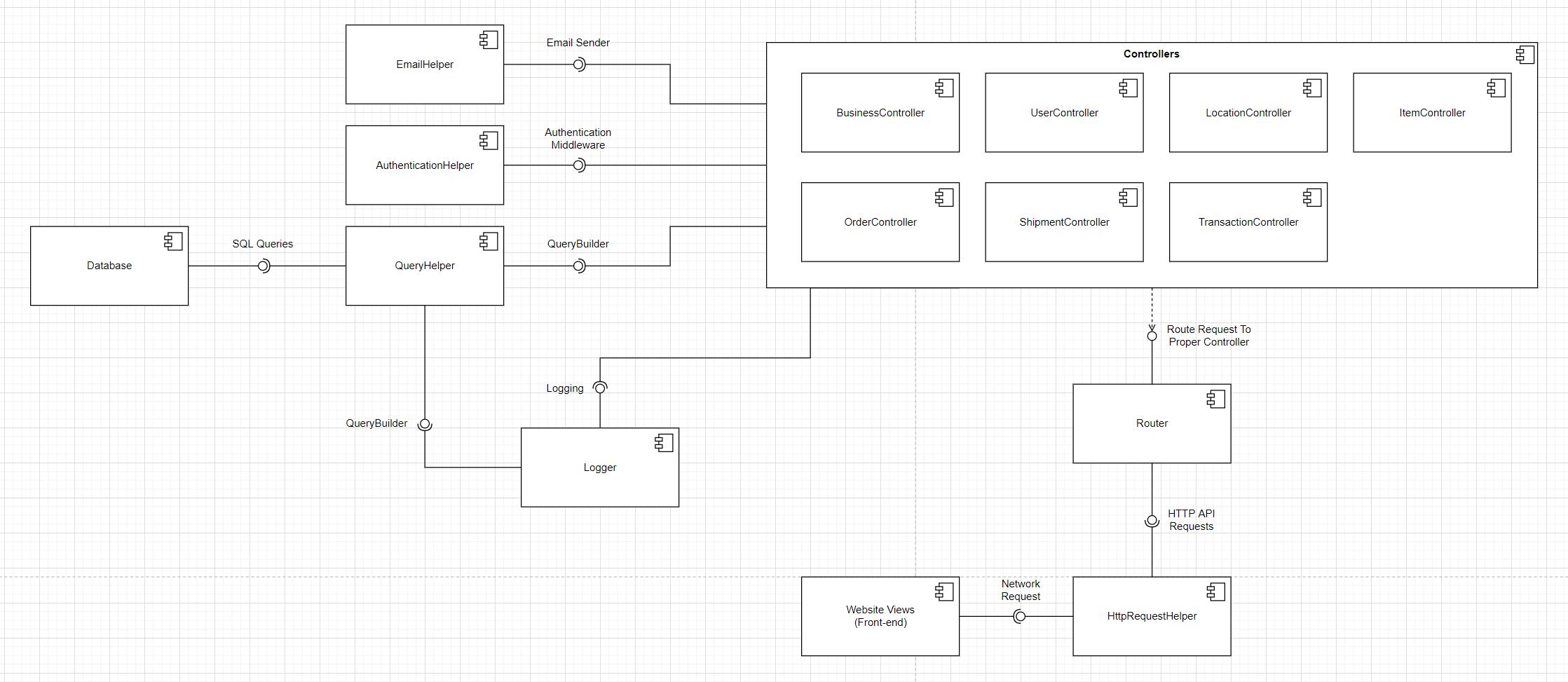
in this view users (administrators, managers and employees) will be responsible for adding in entries to the database and removing ones when they are no longer needed. the employees will be mostly dealing with creating orders, warehouse mangers will be responsible for location specific inventory and managing the incoming and outgoing shipments and the administrators will be responsible for creating new users, businesses, and locations.

# 

**Figure 9:** *Domain model diagram of the system.*

# 5 Physical view

The purpose of the physical view is to describe the system from an implementation standpoint. Its purpose is to allow developers to conceptualize The following component diagram outlines the general architecture of the system. Since the system will use an MVC architecture, the diagram focuses on the implementation of controllers as other aspects are covered in different sections of this document. The system will feature a set of controllers which will manipulate the data stored in various models used by the system (ex: user, item, location). The user interface will interact with an HTTP API to execute various operations on different data models.



**Figure 10:** *Component diagram of the system.*

# 

# 6 Inconsistencies

Currently the only known inconsistency we have is that the frontend is not a pure view. The reason for this is that in order to get some desired behaviors (e.g., automatic logging out) the frontend does need some code that makes it take actions without the user having to perform any actions.

# 7 Bibliography

<https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm>

<https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choice>

<https://scrumguides.org/scrum-guide.html>