SSJ PACKAGE TRACKER

Software Design Document

Team Name
Samuel Anderson
Sanat Bhandari
Jerome Schmidt

Introduction

Purpose

The SSJ Package Tracker offers clients an intuitive interface to track their couriers dispatched via our partner, Bohn's Drones Inc., in Lincoln and Omaha, Nebraska, area. The platform includes a suite of tools you will need at every stage of the Professional Delivery lifecycle.

Design Goals

The SSJ Package Tracker is designed to make it easy and simple for either the customer or a Bohn's drones staff member to observe delivery status and manage drones. It will use a 3-tiered architecture consisting of a database for information pertaining to the drones and packages, an application tier for all necessary logic, and a presentation tier for display relevant information to those using the system.

Design Trade-Offs

Due to time constraints and the unspecific requirements, anyone can log in as a customer or a staff member even though they may not necessarily be a verified customer or a staff member. Another trade off we are making involves not making the design for the front-end more fleshed out, instead opting for a more basic look, in order to raise visual appeal and lessen confusion on the part of the user.

Definitions, Acronyms, and Abbreviations

Delivery Request: The delivery request is generated when either of the clients, the sender or the receiver, initiate the drone activation protocol from their UD

Customer Service Team and Staff: Our dedicated team of hard-working employees to take care of the customer's needs, because the customer comes first!

Site of Operation: Site of Operation refers to the site and its corresponding parameters where the drone is currently at

References

Requirements Analysis Document - TEAM02

Overview

In a nutshell, SSJ Package Tracker is a proprietary software to keep a track of the couriers from the sender-side and the receiver-side. Our software uses a 3-tiered architecture to provide a robust and efficient software experience as well as making additions to the product as easy as possible.

Current Software Architecture

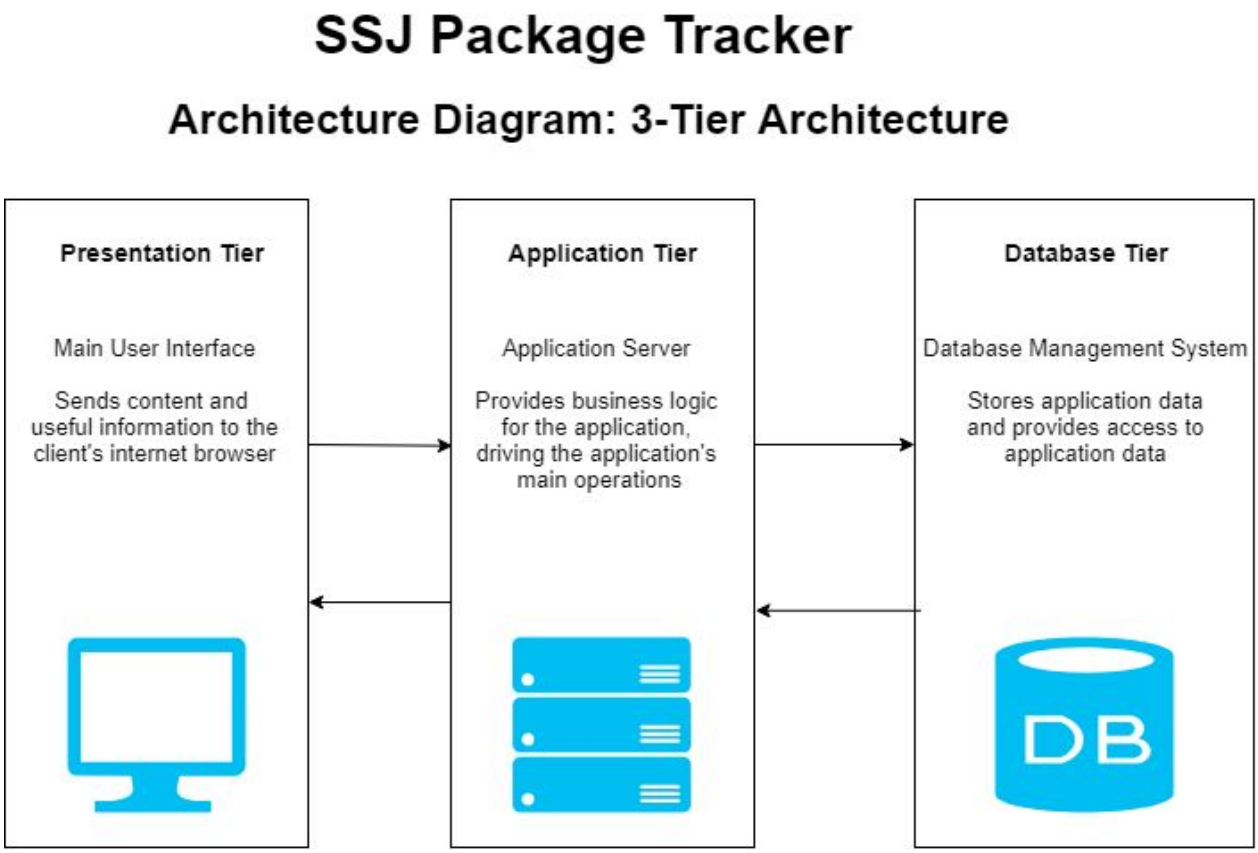
Yada

Proposed Software Architecture

Overview

The SSJ Package Tracking software solution will use a three tier architecture with a Presentation Tier, a Application Tier, and a Database Tier. the Presentation Tier will be the interface the user interacts with, the Application Tier will be where all necessary logic takes place, and the Database Tier is where all information regarding the drones, packages, or customers will be stored.

Subsystem decomposition



Our software system will comprise a 3-tier architectural system for its logical computing. The benefits of a 3-tier architectural system are numerous. One advantage is scalability. By separating the subsystems into 3 different layers, we can scale each layer independently, depending on the need of each layer at any given time. Also, with a 3-tier architecture, our software team's 3 members will be able to specialize in the 3 areas of front-end, server back-end, and database back-end development, allowing our team to develop each subsystem independently in a very efficient manner.

Hardware/software mapping

The only hardware the SSJ Package Tracking System will interact with is the CSCE server that the system's database runs on. Thus, the topic of hardware/software mapping is not very relevant or applicable to our software system.

Persistent data management

The SSJ Package Tracking System will use a relational database run on the CSCE servers to track and manage the drones and packages being transported. The database will be updated anytime there is a change in status of either a package or a drone.

Access control and security

The SSJ Package Tracking System does not do any forms of login authentication or encryption, but the system will handle keys in terms of tracking numbers for databasing purposes.

Global software control

A request is initiated by the user in the Presentation Tier, which will then pass through the Application Tier where any logic will be performed and then passed to the Database Tier where any and all necessary information will be present. To avoid any concurrency issues, the database will be updated as soon as possible so that all information in the database is as accurate as possible.

Boundary conditions

On each startup of the system, any updates to the database content will be saved as soon as possible in order to mitigate any issues that may arise from an unexpected system crash, ex: critical customer delivery data.

Subsystem Services

Database Tier:

- Accepts queries from the Application Tier and sends results of the queries back to the Application Tier
    - The Database Tier takes orders from the Application Tier, and the database is updated accordingly. Information from the database is then sent back to the Application Tier.
    - Ergo, the Database Tier does not have methods, because it only consumes services and sends the results back to the Application Tier.

Application Tier:

- Accepts inputs from the Presentation Tier and passes queries generated from the inputs to the Database Tier
    - updateDroneData(): updates data relating to Drone objects in the database
    - updateOrderData(): updates data relating to Information objects in the database
- Accepts the outputs of the Database Tier, performs any logic required, and passes results generated from the logic to the Presentation Tier. In other words, processes information from the database, and returns this information to the Presentation Tier in a human-readable format
    - getDroneList(): obtains the list of drones as well as their status
    - getOrderList(): obtains the list of orders as well as their status
    - displayDroneList(): displays the list of drones to the Staff user
    - displayOrderList(): displays the list of orders to the Customer or Staff user

Presentation Tier:

- Takes input from the user and passes that on to the Application Tier
    - generateDeliveryRequest(): Generates a delivery request for a drone at the customer's location

    - dispatchEmptyDrone(): At the request of a staff user, dispatches a drone from one depot to another, or to a customer's location after they have submitted a delivery request
- Receives output from the Application Tier and displays outputs to the user
    - getDroneList(): Returns the list of all drones in the system
    - checkDeliveryStatus(): Returns the delivery status of a customer's order
    - checkPackageInformation(): Returns the information of a customer's package, including its time of dispatchment, origin address, delivery address, package tracking number, etc.

Class Interfaces

Portal:

- Serves as the primary class that works with the front-end interface. Contains methods related to logging in and some methods that pass information and requests to the Application class.
    - Login lets a user log in as either a customer or staff
    - Logout logs the user out and returns them to the home screen

Application:

- Contains methods that both retrieve information from the database and update the database with updated information
    - getDroneList gets a list of drones as well as their status
    - getOrderList gets a list of orders as well as their status
    - displayDroneList displays the drone list to user
    - displayOrderList displays the order list to user
    - dispatchEmptyDrone lets a staff member dispatch an empty drone to a different depot
    - updateDroneData lets the application tier update drone location and/or status as necessary
    - updateOrderList lets the application tier update the order list as necessary

Database:

- The data that is crucial to system operations. This data is interacted with and manipulated by the Application business logic subsystem
    - droneList is the list of all Drone objects in the system
    - orderList is the list of all Information objects in the system

Drone:

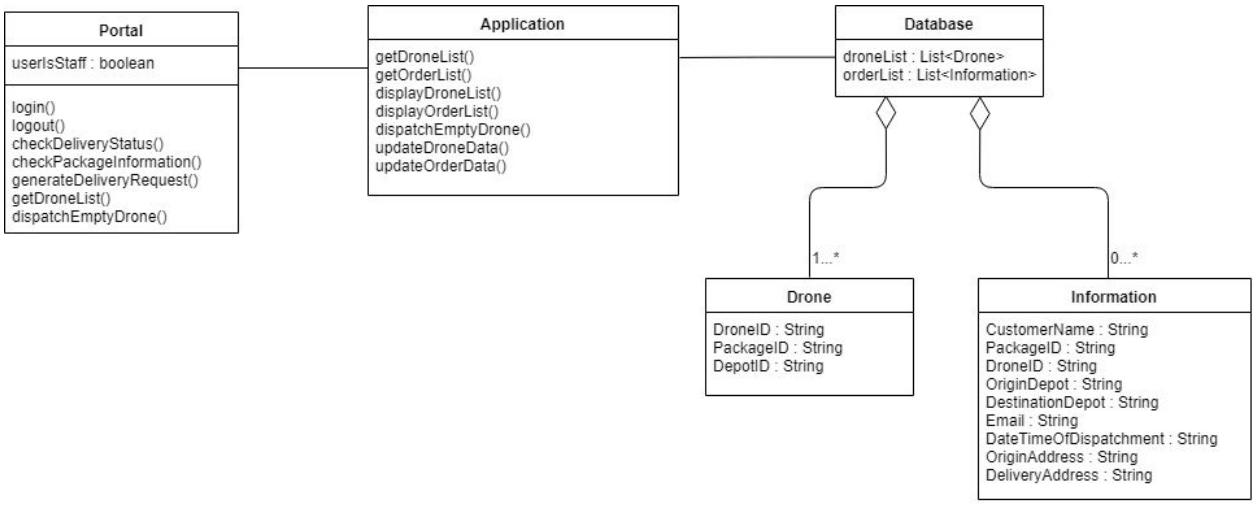
- Database table that keeps track of the DroneID, PackageID, and DepotID fields for Drone objects in the system.
    - DroneID is the drone's identification number
    - PackageID is the package tracking number of the package that the drone is currently transporting (if it has one)
    - DepotID is the identification number of the depot that the drone is currently residing at (if it is at one)

Information:

- Database table that keeps track of the CustomerName, PackageID, DroneID, OriginDepot, DestinationDepot, Email, DateTimeOfDispatchment, OriginAddress, and DeliveryAddress fields for Information objects in the system.
    - CustomerName stores the name of customer that has ordered the package
    - PackageID stores the identification number of a given package making it easier to keep track of
    - DroneID is the identification number of the drone that is currently transporting the package (if the package is currently being transported by a drone)
    - OriginDepot is the last depot that the drone carrying the package was at
    - DestinationDepot is the depot that the drone carrying the package is on its way to
    - Email is the email address associated with the customer that has ordered the package
    - DateTimeOfDispatchment is the date and time of the delivery being dispatched, so the time that the first drone was sent off to pick up the package from the customer's delivery location
    - OriginAddress is the address that the package originated at for the delivery
    - DeliveryAddress is the address that the package will ultimately end up at by the end of the delivery

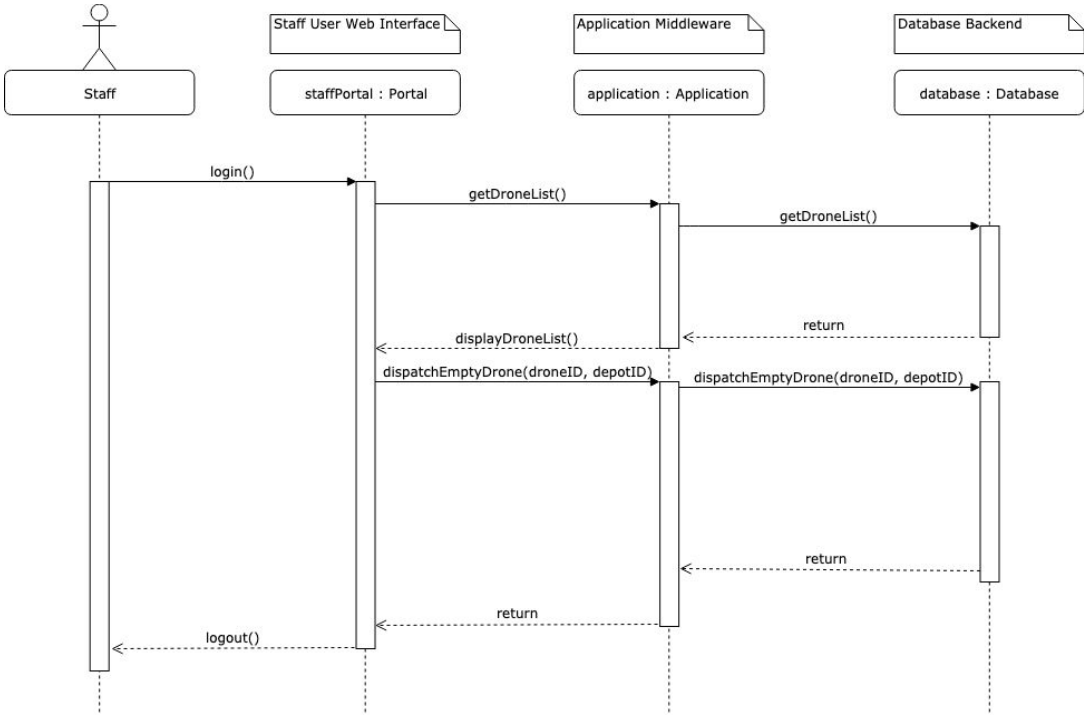
Detailed Design

Class Diagram

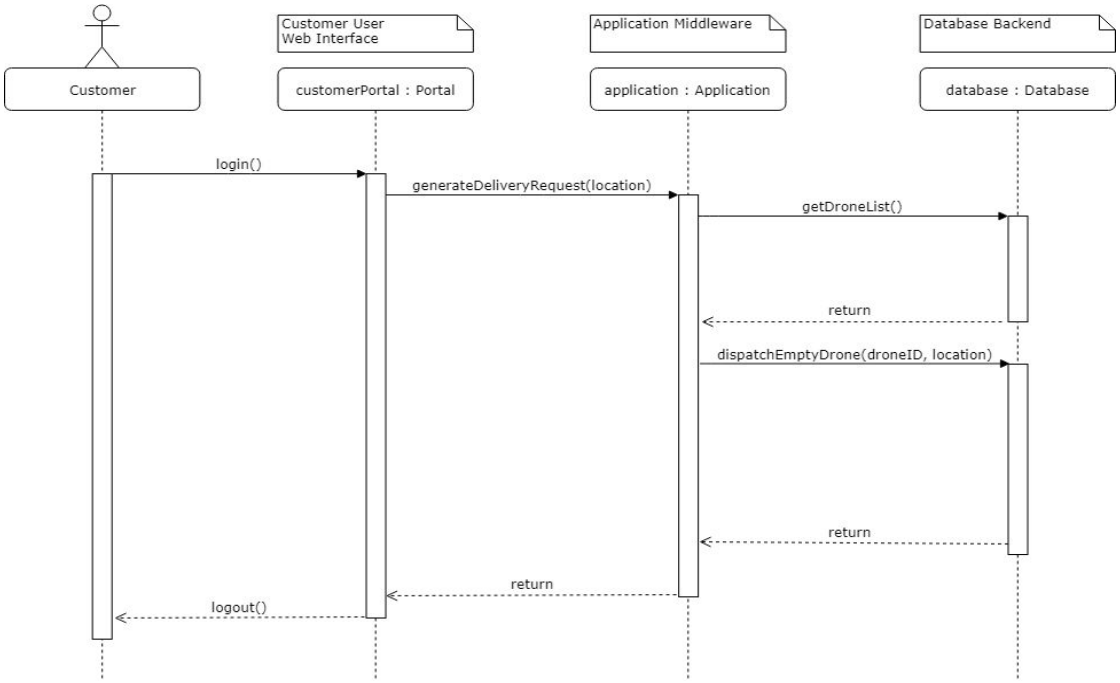


Sequence Diagrams

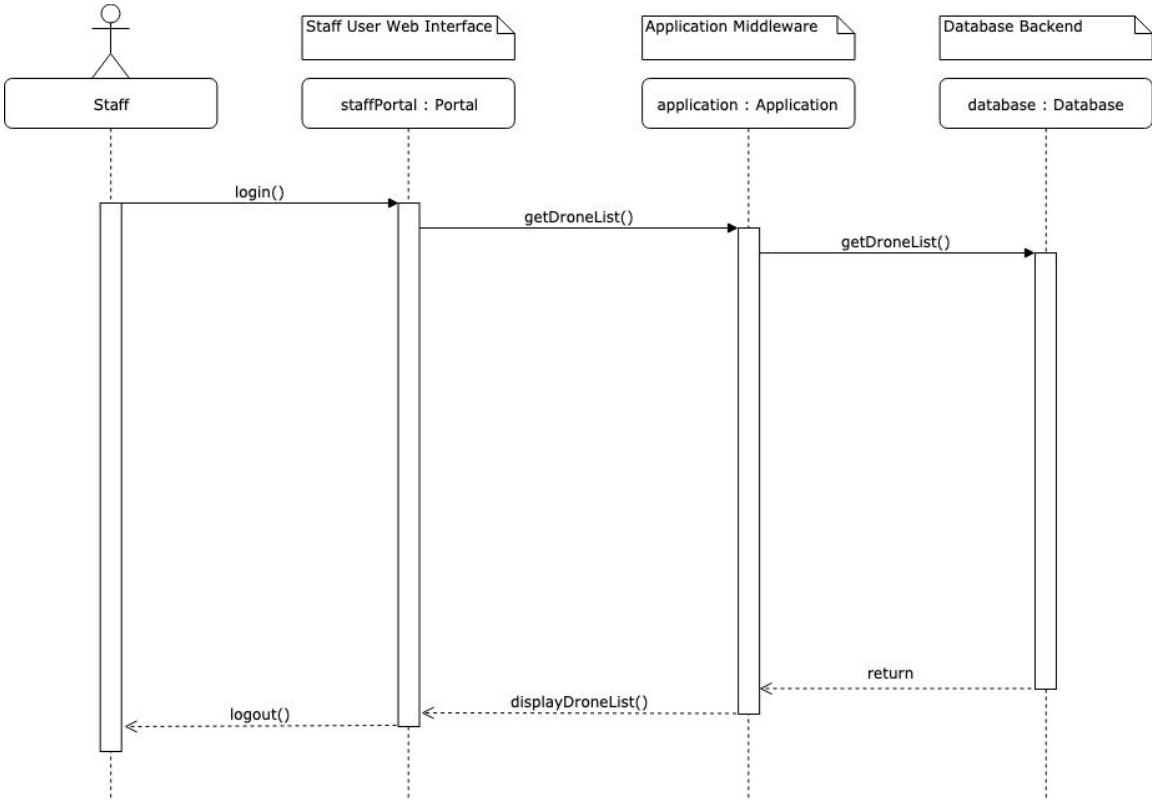
Dispatching Empty Drones



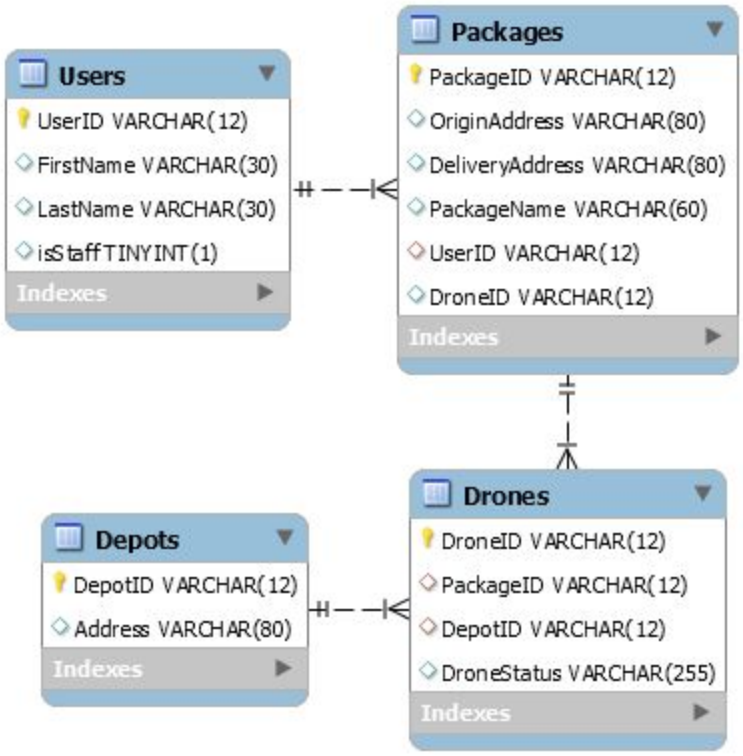
Generating Delivery Request



Requesting Drones List



BONUS: Database ER Diagram



Glossary

SSJ Package Tracker - Samuel, Sanat, & Jerome Package Tracker