Modernizing Hiko Sandwich:

A Python-Based Proof of Concept for Streamlined Ordering

Course: KAN-CDSCO2402U Programming, Algorithms and Data Structures
Programme: MSc in Business Administration and Data Science
Student number: 152463

Physical pages: 15
Characters: 23747
Date of submission: 20/12-2024

Github: https://github.com/myokom/prog-exam-cbs
Link to web app: https://prog-exam-cbs.streamlit.app

Table of Contents

Abstract

This project presents a Python-based proof of concept (POC) system designed to enhance the customer experience and operational efficiency at Hiko Sandwich, my go-to sandwich shop. The system addresses key challenges such as long wait times, a complex ordering process, and an ineffective loyalty program by introducing features like streamlined ordering, a modernized loyalty system, and cost-effective online order management. By leveraging object-oriented design, modular programming, and tools like Streamlit, Pandas, and Plotly Express, the solution offers a user-friendly interface for both customers and administrators. Key functionalities include user account management, customizable sandwich orders, student discounts, and an admin dashboard for inventory and customer management and analytics. While this POC is not production-ready, it demonstrates the feasibility and potential value of such a system, providing Hiko Sandwich with insights for decision-making regarding further development.

Keywords: *Python, customer experience, Proof of Concept (POC), Streamlit, operational efficiency*

Introduction

Problem

Recently, while studying with friends at CBS BitLab, we decided to grab lunch at our favorite spot, Hiko Sandwich. The sandwiches are fantastic, and we always appreciate the freshness, quality and generous size. However, during our visit, we noticed a few recurring issues that impact the overall experience. These challenges often arise, especially during the busy lunch hours.

Lunchtime Chaos

Queue

At lunchtime, we are clearly not the only ones who think of Hiko Sandwich. When we arrived, there were already a few customers ahead of us. As we began placing our orders, another group

of four entered the small sandwich shop, adding to the growing queue. During peak hours, the limited space feels even more cramped, making the atmosphere tense for those at the counter trying to finalize their sandwich choices

Ordering

Ordering sandwiches for a group of five takes time – not because the staff is slow but because the process requires extensive customer involvement. Hiko's customizable menu, while amazing, presents nearly infinite choices. Customers must first choose from two types of bread, followed by four different spreads. Next, they select one of five proteins and can then add their choice of up to 15 vegetables. To top it off, there are six dressings to choose from, along with optional extras like avocado, cheese, and bacon.

These options amount millions of options, which can overwhelm customers and slow down the ordering process. Each decision adds time, and indecisive customers further prolong the wait for everyone – first-time customers, unfamiliar with the menu and options, are often even slower. Though effective in larger chains like Chipotle, this approach is harder to sustain at Hiko, where typically only one or two employees are on duty.

Lost Loyalty Cards, Lost Rewards!

Hiko Sandwich recognizes the value of a loyalty program, offering a stamp card where the 10th sandwich is free. However, the system has notable drawbacks. Despite visiting Hiko regularly for over three years of studying at CBS, neither I nor anyone I know has ever completed a stamp card. The issue lies in the card itself – it is small, easy to lose, and often forgotten, leaving customers with multiple partially filled cards. And let's face it – what is a student's favorite thing? Free food! Yet this loyalty program, as it stands, makes it surprisingly difficult to achieve that reward.

Convenience at a Cost

This might leave you wondering: why not just order online? While it is possible through Wolt, the convenience comes at a steep price. The same sandwich I paid 77 DKK for today would cost

104 DKK on Wolt, and delivery is mandatory – an unnecessary expense when Hiko Sandwich is just a two-minute walk from CBS BitLab (Wolt | Hiko Sandwich, n.d.). Plus, ordering through Wolt means I can not even get a stamp on my loyalty card! It is not ideal for Hiko either, as Wolt takes about 30% in commission, leaving Hiko with even less than normal (Thykier, 2024).

Objective

The objective of this project is to enhance the customer experience and operational efficiency at Hiko Sandwich by develop a Python-based system that streamlines ordering, reduces wait times, modernizes the loyalty system, and enables cost-effective online orders for pickup, the project aims to improve customer satisfaction, increase operational efficiency, and foster loyalty.

A fully operational system ready for implementation is beyond the scope of this exam, as it would require several additional technical enhancements. These include implementing robust user authentication and role management, as well as integrating a scalable data storage solution, such as a relational database like PostgreSQL, hosted on the cloud. These features are essential for a production-ready system but fall outside the focus of this project.

Thus, the objective of this project is to deliver a proof of concept (POC) that allows Hiko Sandwich to evaluate the potential value of such a system. This POC aims to support their decision-making process by helping them determine whether they should pursue an off-the-shelf solution, invest in having someone build a custom system, or continue with their current operations.

Requirements Analysis

Functional Requirements

User functionalities

User Account Management

- Create Account: The system must allow new users to register by providing a Customer ID, Name, Email, and Phone - for testing purposes Password has yet to be included.

- Login: The system must allow returning customers to log in using their Customer ID.
- Update Contact Information: The system must provide a function `update_contact_info` enabling users to update their email and/or phone number at any time.

User Types and Discounts

- Regular Users: By default, the system treats users as `RegularUser` objects with no automatic discounts.
- Student Users: If a user's email ends with @student.cbs.dk, the system must treat them as a `StudentUser`. Such users receive a 5% discount on their total order cost, implemented in the `apply_discount` method.
- Admin Users: The system must support an `AdminUser` type who can view all orders, view all customers, and manage the inventory.

Order Placement and History

- Create Orders: Logged-in users must be able to initialize a new order. Each order should have a unique order_id and a reference to the customer placing it.
- Add Sandwiches to Order: Users must be able to add multiple `Sandwich` objects to an order. The order should store these sandwiches and keep track of their ingredients.
- Calculate Total Cost: The system must be able to calculate the total cost of an order using the `calculate_total` method. The calculation should consider:
    - Base Pricing Based on Time: If the order is placed between 8:00 and 14:00, the base price is 77 DKK; otherwise, it is 80 DKK.
    - Extra Ingredients Costs: The system must add extra charges for selected extras using the `get_extra_cost` method from Inventory.
    - Loyalty Program: Every 10th sandwich is free. The system must determine the number of free sandwiches earned using the Loyalty class's `free_sandwiches_earned` method and apply those discounts to the order.
    - User-Specific Discounts: If the user is a `StudentUser`, the system must apply a 5% discount to the final total.

- Update User History and Counts: Once an order is placed, it must be recorded in the user's order history, and the total sandwich count for that user should be updated. Users can retrieve their past orders via `get_order_history`.

Ingredient Selection and Validation

- Select Ingredients: The system must allow `Sandwich` objects to be customized by selecting bread, spread, protein, vegetables, dressing, and extras.
- Validation of Ingredients: Before finalizing a sandwich, the system must validate that chosen ingredients are available and valid. Methods like `is_valid_bread` and `is_valid_vegetables` ensure only permissible ingredients are included.

Admin functionalities

Inventory Management

- Add/Remove Ingredients: Admin users must be able to add new ingredients or remove existing ones using `manage_inventory`. This includes specifying a category (e.g., "bread", "extra") and a price for newly added ingredients.
- No Removal of Mandatory Placeholders: The system must prevent removal of placeholder ingredients like "No spread" or "No extras" to maintain a consistent state.
-

Order Viewing

- View All Orders: Admins must be able to retrieve and view all orders currently in the system. If no orders exist, an appropriate message should be shown.
- View All Customers: Admins must be able to view the list of all customers, enabling oversight of user activity and data.

Non-functional Requirements

As this project serves primarily as a POC, the non-functional requirements are scoped to demonstrate basic viability and user experience rather than to meet production-ready standards.

Security

During the Proof of Concept (POC) phase, the system will operate within a controlled test environment, making extensive security measures less critical. However, as the project matures and potentially moves toward a production environment, robust security practices (e.g., encrypted communication, stronger authentication methods, and protection against common vulnerabilities) will become a top priority.

Performance

As the POC also serves as a demonstration tool, the system should run smoothly and respond within a reasonable timeframe. Minimal delays and basic responsiveness are expected, ensuring that potential stakeholders can experience a stable and efficient ordering process without distractions caused by slowdowns or instability.

User Experience

The user interface should be intuitive enough for both Hiko Sandwich's staff and test customers to navigate easily. Clear and straightforward feedback, including simple error messages, will help users understand any issues and correct their actions quickly. While comprehensive UI/UX refinements may not be essential at this stage, the POC should establish a positive baseline user experience to build upon.

Extensibility

The codebase should be well-structured, with clear comments and logical organization. This makes it easier for developers who may join the project later to understand and modify the code quickly. As Hiko Sandwich considers scaling this proof of concept into a fully developed,

production-ready system, focusing on clean architecture and maintainable code from the start will be invaluable.

System Design

Architectural Design

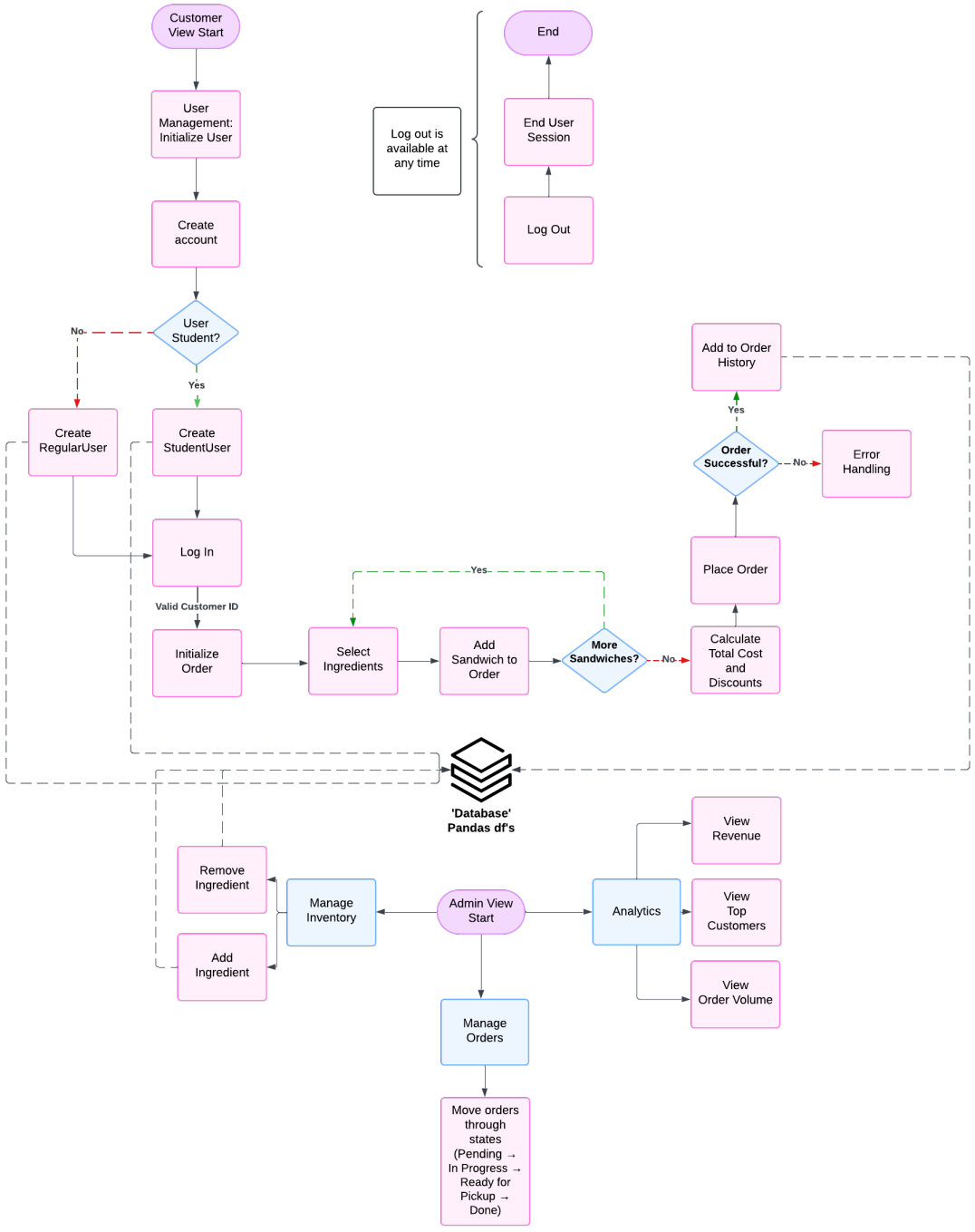

Figure 1: Diagram of the architectural flow

Module Design

At the foundation of the application is the `User` class, which serves as a base class for all current and future user types in the system. This class defines common attributes such as *user_id*, *name*, *email*, and *phone*. It also provides methods like `update_contact_info`, enabling users to update their email and/or phone number, and `apply_discount`, which by default offers no discount, however, specialized user types will override this method to provide targeted discount logic.

Building upon the `User` class, the `RegularUser` subclass introduces customer-specific functionality. It maintains an `order_history` list and tracks how many sandwiches the user has purchased. The methods `add_order` and `get_order_history` helps to keep track of the user's past purcheses - important for the loyalty program. Extending `RegularUser`, the `StudentUser` subclass provides a specialized discount mechanism. By overriding `apply_discount`, it grants a 5% discount if the user's email ends with *@student.cbs.dk*. This subclass inherits all the order management and tracking capabilities of `RegularUser` while adding its own discount logic. Another specialization of `User` is the `AdminUser` subclass, designed for administrative tasks. It provides methods such as `view_all_orders`, `view_all_customers`, and `manage_inventory`.

The `manage_inventory` method leverages the `Inventory` class, granting admins the ability to add or remove ingredients as needed. The `Inventory` class, upon initialization, sets up default price lists for various categories like breads, spreads, proteins, vegetables, extras, and dressings. It also includes methods like `add_ingredient` and `remove_ingredient` for dynamically updating these offerings, ensuring the menu can adapt over time. Validation methods such as `is_valid_bread` and `is_valid_spread` guarantee that only legitimate ingredients are used in sandwich creation.

Beyond user-specific discounts, the application offers a broader loyalty program for all regular customers. This is managed by the `Loyalty` class, which currently implements a "10th sandwich free" promotion. Its `free_sandwiches_earned` method calculates how many

complimentary sandwiches should be applied to an order, based on the total number of sandwiches a customer has previously purchased. Future loyalty perks and promotions can be integrated here as the system evolves.

The heart of the purchasing process lies in the `Sandwich` and `Order` classes. A `Sandwich` object encapsulates ingredient selections, including breads, spreads, proteins, vegetables, dressings, and extras. Through methods like `select_bread`, `select_protein`, and `add_vegetables`, it ensures that each chosen ingredient is valid and available. The `get_price` method then calculates the cost of the sandwich by combining a base price (determined at order time) with any added extras.

Finally, the `Order` class brings everything together. An order consists of one or more `Sandwich` objects placed by a particular User. The Order class offers methods to `add_sandwich`, `calculate_total`, and determine the time-based base price through `get_time_based_price`. It also integrates with `User` and `Loyalty` features, applying both user-specific discounts and loyalty-based free sandwiches. This ensures that every purchase is accurately priced, tracked, and easily reviewed later on, supporting a seamless and transparent purchasing experience for both customers and administrators.

Implementation

Tools and Technologies

```
from datetime import datetime
```

The `datetime` module is used to enable the application to handle dates and times effectively (Python Software Foundation, n.d.). This functionality is important as sandwich prices are different depending on the time of day. Additionally, each order is timestamped with the date and time, allowing for detailed analysis. This data allows Hiko Sandwich to identify peak hours and busy days, providing valuable insights that can help to optimize worker scheduling and improving operational efficiency.

`import pandas as pd`

While the `pandas` library is often used for data manipulation, it is in the case mostly used as a mock database that can be used for querying or visualizations (Pandas Documentation, n.d.). The data in the `pandas` data frames does not need further manipulation as the data is mockup data created through a random simulation of orders. This is because the app has yet to face real-world data, which is often messy and requires cleaning and preprocessing before use.

`import plotly.express as px`

To gather insights from the 'database' and deliver clear data visualizations to Hiko Sandwich, I chose to use Plotly Express. Although I often rely on Matplotlib for data visualization, Plotly Express stood out for this project due to its user-friendly, interactive visualizations that seamlessly integrate with web-based applications (Plotly, n.d.). This makes it an ideal choice for the admin view, where analytics will be presented in a dynamic and accessible format.

`import streamlit as st`

Code in a notebook is often not much more than an idea, and therefore it can be challenging for non-technical stakeholders, like Hiko Sandwich, to fully grasp its potential or decide whether to commit to building out the system. While software engineers or data scientists might easily envision how the idea could be brought to life, it remains abstract for others. To bridge this gap, I used Streamlit, an open-source Python framework with a user-friendly API that allows for rapid development of web applications without requiring front-end expertise (Streamlit, n.d.). With this I could quickly move from a Jupyter Notebook to a POC that Hiko Sandwich can interact with directly, enabling them to see and evaluate its value firsthand.

Code Structure

- `helper_functions/`
    - `classes.py`: Defines key classes used throughout the project, such as Order, Customer, or Ingredient to facilitate object-oriented design.

        - `update_dfs.py`: Contains functions to update and manipulate the data frames, ensuring efficient handling of the mock database.
- `simulated_data/` (mock data used for testing and demo)
    - `customers.csv`: Simulated customer data, such as customer names or IDs.
    - `ingredients.csv`: A list of available sandwich ingredients with details like names, categories, or prices.
    - `orders.csv`: Mock order data, representing transactions processed by the system.
- `app.py` The main Python script that runs the web app
- `notebook.ipynb` Notebook for the creation of the classes and testing them.
- `simulated_data.ipynb` Notebook for generating and simulating mock data
- `requirements.txt` Python dependencies needed to run the web app.

    This is required for Streamlit Cloud to know what libraries is necessary to make the pap run. Thus, to run this locally all you need to do is to open a terminal, go to the directory folder and run:

    ```
    pip install -r requirements.txt
    streamlit run app.py
    ```

Key Code Snippet

```
# Base Class
class User:
    def __init__(self, user_id, name, email, phone):
        """
        Initialize a new User object with the parameters: user_id, name, email,
and phone.
        """
        self.user_id = user_id
        self.name = name
        self.email = email
        self.phone = phone

    def apply_discount(self, total_cost):
        """
        Default: no discount for a regular user.
        Override in subclasses for specific discount logic.
        """
        return total_cost  # Default: no discount
```

```
    def __str__(self):
        """
        Formatted string representation of the user.
        """
        return f"User ID: {self.user_id}\nName: {self.name}\nEmail:
{self.email}\nPhone: {self.phone}"

# Subclass: Student User
class StudentUser(User):
    def __init__(self, user_id, name, email, phone, domain="@student.cbs.dk",
discount_rate=0.05):
        super().__init__(user_id, name, email, phone)
        self.student_domain = domain
        self.student_discount_rate = discount_rate

    def apply_discount(self, total_cost):
        """
        Apply a 5% discount if the user's email ends with the student domain.
        """
        if self.email.lower().endswith(self.student_domain):
            return total_cost * (1 - self.student_discount_rate)
        return total_cost

    def __str__(self):
        return super().__str__() + "\nStatus: Student"
```

Results and Discussion

Results

The POC web app can accessed through this link: https://prog-exam-cbs.streamlit.app

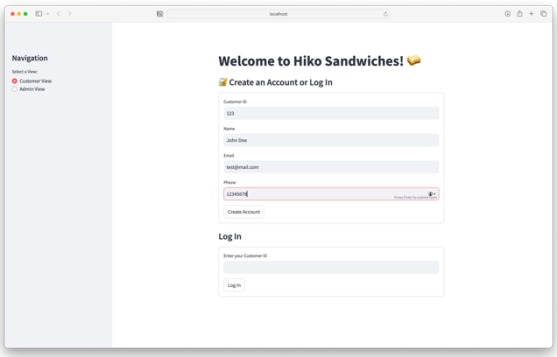

Figure 2: The "Create an Account and Log In" page, displaying both the registration form for new users and the login fields for returning users.

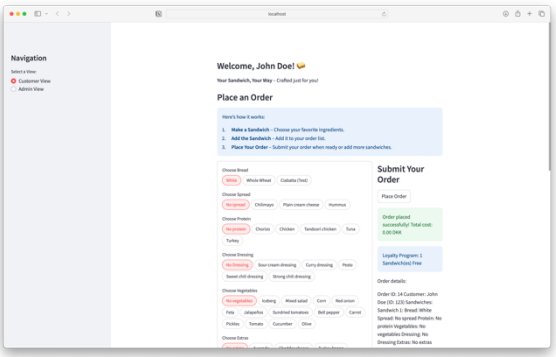

Figure 3: The "Place an Order" interface, allowing customers to build custom sandwiches by selecting ingredients from various categories and then submit their orders. The order has been placed and registered in the customers purchase history and in Hiko Sandwiche's database. The total cost is 0 DKK because it is the customer's 10th sandwich

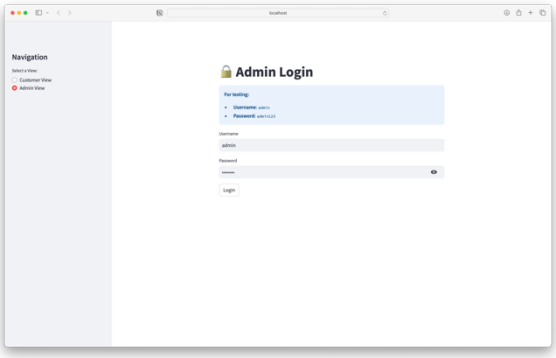

Figure 4: Log in page for the admin view

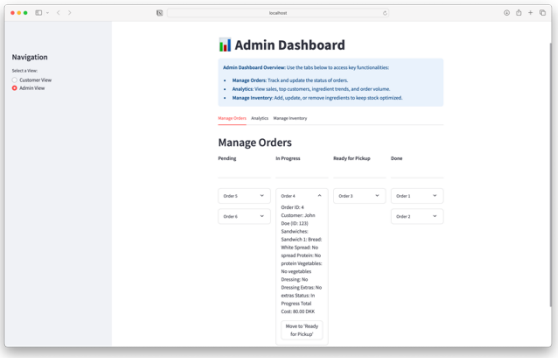

Figure 5: Log in page for the admin view

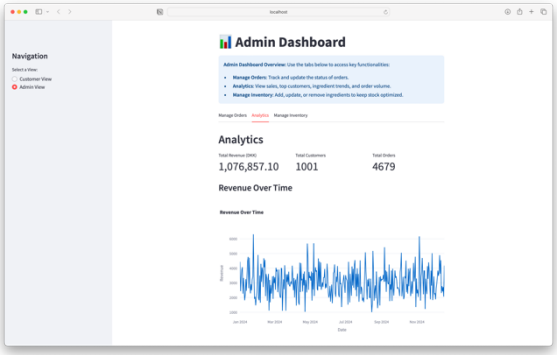

Figure 6: The Admin Dashboard overview, showing key performance metrics such as total revenue, total customers, and orders, along with revenue-over-time analytics

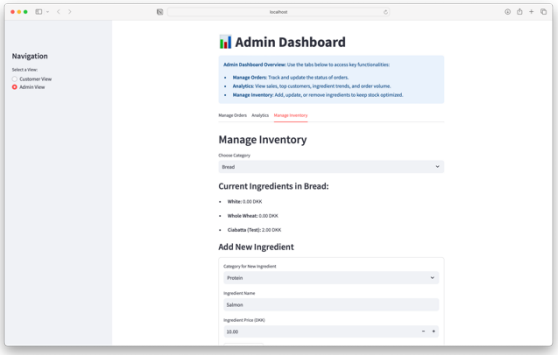

Figure 7: The "Manage Inventory" section of the Admin Dashboard, enabling admins to add, update, or remove ingredients.

Discussion

After visiting Hiko Sandwich and identifying key issues, I began coding by building the necessary classes. However, I realized that presenting the system in a Jupyter Notebook or command-line interface would not clearly convey its value to Hiko Sandwich. To solve this, I used Streamlit, a Python framework that quickly transforms notebook-based work into interactive web applications. In the `simulated_data.ipynb` file, I generated a year's worth of mock data for testing purposes. Additionally, I tested various sorting algorithms to determine the best option. The results showed that Python's and Pandas' built-in sorting functions were sufficient for this proof of concept.

Bubble sort	0.08889 seconds
Merge Sort	0.00120 seconds
Quick Sort	0.00062 seconds
Built-in sorted()	**0.00011 seconds**
Pandas' sort_values()	**0.00056 seconds**

Figure 8: Results from sorting algorithm testing

Conclusion

This project presents a proof of concept (POC) for a Python-based system aimed at enhancing the customer experience and operational efficiency of Hiko Sandwich. By addressing key challenges such as long wait times, complex ordering processes, and an ineffective loyalty program, the proposed solution demonstrates how technology can streamline operations and deliver value. The system leverages modular programming, object-oriented design, and tools like Streamlit, Pandas, and Plotly Express to provide a functional, user-friendly interface for customers and administrators. Features such as time-based pricing, loyalty rewards, and student discounts showcase the flexibility and potential of the solution. Additionally, the mock database and interactive visualizations enable Hiko Sandwich to analyze order trends and peak hours, offering insights for better planning and resource management.
While this POC is not yet production-ready, it lays a solid foundation for further development.

Future Work

While the path to a production-ready system is long, the next iteration of the app should focus on implementing key features that will bridge the gap between this proof of concept and a fully operational solution. These enhancements include:
Robust User Authentication and Role Management: Implementing secure login systems with password encryption and role-based access control for customers and administrators.
Scalable Data Storage: Replacing the current mock database with a relational database like PostgreSQL, hosted on the cloud, to ensure scalability and data persistence.

Learning objectives

- ☆ *Understand fundamental Python programming concepts, techniques and methods.*

  To show my understanding of Python fundamentals I used variables, data structures (dictionaries, lists, and data frames), conditionals, loops (implicitly through Streamlit components and data processing), and functions. Moreover, I implemented classes to represent entities like `User`, `Order`, and `Inventory`.

☆ *Design and implement programs using Python programming language using its appropriate syntax and features.*

☆ *Demonstrate understanding of imperative, declarative and object-oriented language features of Python language and know when it is appropriate to use each.*

I demonstrate my understanding of ***imperative features*** through my use of Streamlit, which relies heavily on `if` statements and explicit control flow to determine what content to display based on user input or conditions. I demonstrate my understanding of ***declarative features*** by using frameworks like Streamlit, pandas, and plotly. Streamlit allows me to declare UI components such as `st.title()` and `st.plotly_chart()` without managing rendering. With pandas, I perform data manipulations like grouping and summing using `groupby()` and `sum()`

Object-oriented features

I demonstrate my understanding of ***inheritance*** with the `RegularUser` class which inherits from `User`. This means `RegularUser` gets access to all attributes and methods from `User`. The `StudentUser` further inherits from `RegularUser` and adds student-specific behavior (e.g., applying a 5% discount). I demonstrate my understanding of ***composition*** in the `Order` class which as a list of `Sandwich` objects - it contains sandwiches without inheriting from the `Sandwich` class. I demonstrate my understanding of ***encapsulation*** in the `Inventory` class by restricting direct access to ingredient data and providing controlled methods like `add_ingredient` and `remove_ingredient`.

☆ *Write programs in Python programming language that make use of external libraries, APIs, etc.*

As described in the Tools and Technologies section I have integrated multiple external libraries. More specifically: `Streamlit` for the web UI, `pandas` for data handling, and `plotly.express` for data visualization

☆ *Demonstrate a basic understanding of algorithm analysis and program complexity.*

I demonstrate my understanding of algorithm analysis in the `simulated_data.ipynb` file where I test different sorting algorithms.

References

European Union. (2016). *General Data Protection Regulation (GDPR)*. EUR-Lex. https://eur-lex.europa.eu/eli/reg/2016/679/oj

pandas. (n.d.). pandas documentation. pandas. https://pandas.pydata.org/docs/

Plotly. (n.d.). Plotly Express. Plotly. https://plotly.com/python/plotly-express/

Python Software Foundation. (n.d.). datetime — Basic date and time types. Python Documentation. https://docs.python.org/3/library/datetime.html

Streamlit. (n.d.). Docs: API reference. Streamlit. https://docs.streamlit.io/develop/api-reference

Thykier, M. (2024, May 17). *Restauranter råber op: Er du klar over, hvor mange penge der går til Wolt?* Politiken. https://politiken.dk/danmark/forbrug/art9893216/Restauranter-r%C3%A5ber-op-Er-du-klar-over-hvor-mange-penge-der-g%C3%A5r-til-Wolt

Wolt | Hiko Sandwich. (n.d.). https://wolt.com/da/dnk/copenhagen/restaurant/hiko-sandwich