Development of a Point-Of-Sale (POS) and Database Software System

Eliya Pelton

Professor Gregory Baumgardner

Department of Engineering

Abstract

This is a software engineering project to develop the software for a cloud-based Point-of-Sale (POS) and database application that can be accessed online from several devices.

Introduction

Purpose

The purpose of this project is to fulfill the needs of the customer, The Gold and Silver Exchange, LLC. The Gold and Silver Exchange is a small business that buys and sells gold, silver, and other precious metals in the form of bullion, coins, jewelry, and other items. Still run by the original founder, the business has not updated its mode of operations since its establishment in 1985.

Background

Currently, employees conduct transactions using hand-written, carbon-paper invoices. There are several problems with this method.

First, it is very time-intensive. Employees must manually write all relevant information down, including an itemized list of what is being bought or sold. For very large transactions, these invoices can span multiple pages, requiring a copious amount of time. Moreover, if the company is conducting a purchase, then the employee must write down the customer information. This consists of the customer's name, address, date of birth, ID or passport number, and the ID or passport expiration date.

Second, these manual invoices are highly prone to clerical error. The employees must calculate the price of each item by looking up the current spot price (which fluctuates every few minutes) and adding the company's premiums. Then, the employee must sum all line item totals. With paper invoices, there is no computer to double-check the employee's calculations, so if the employee makes a mistake, then it is propagated through completion of the transaction. This will either result in a loss of money for the company or for the customer, which in turn results in an unprofessional company reputation. Figure 1b illustrates an example of an invoice with clerical mistakes.

The Gold and Silver Exchange, LLC

1800 Sunny Rd, Long Beach CA *(707) 777-7777*

Name: John Carretini	Date: 10/02/2023
Address: 501 Parkview Lane, Santa Barbara, 95321	
ID: C3501776	Expiration: 05/19/2025
Phone: (707) 810-9325	DOB: 05/19/1972
Payment Method: Check #1302	
Silver Spot: $21.03	Gold Spot: $1850.34

Qty	Item	Price	Total
2	1/4oz Gold Krugerrand	$531.97	$1,063.95
1	1g Gold Bar		$66.64
10	1oz Silver Rounds	$25.03	$250.30
1	100z Silver Bar		$240.30

Customer signature:	John Carretini	Subtotal: $1,621.18
		Total: $1,771.14

Figure 1a: Correct Invoice

The Gold and Silver Exchange, LLC

1800 Sunny Rd, Long Beach CA *(707) 777-7777*

Name: John Carettini Carretini	Date: 10/02/2023
Address: 501 Parkview Lane, Santa Barbara, 95321	
ID: C3501776	Expiration: 05/19/2025
Phone: (707) 810-9325	DOB: 05/19/2025 1972
Payment Method: Check omitted check number "#1302"	
Silver Spot: $20.76 $21.03	Gold Spot: $1845.07 $1850.34

Qty	Item	Price	Total
2	1/4oz Gold "Krugerrand"	$530.46	$1,060.92
1	1g Gold Bar		$66.45
10	1oz Silver Rounds	$23.76	$237.60
1	100z Silver Bar		$237.60

incorrect addition of premium:
{spot} + {premium} = {price}
$20.76 + **$4** = **$24.76**

Customer signature:	John Carretini	Subtotal: $1,602 $1602.56
		Total: $1,750.19

Figure 1b: Invoice with clerical Errors

In the example, the employee misspells the customer's name, accidentally copies the expiration date into the Date-Of-Birth field, fails to record the check number, and omits the word "Krugerrand" from the item description so that later readers will not be able to identify what item

was sold. The employee uses the wrong silver and gold spot prices, causing all subsequent prices to be lower than they should be. The employee also incorrectly calculates the price of a 1oz Silver Round by adding a premium of $3 rather than $4. Lastly, the employee mistakenly sums the line items as $1,602 instead of $1,602.56. After all these calculation errors, the employee charges the customer a total of $1,750.19 when the total should have been $1,771.14.

Because the company has no formal inventory system, the owners rely on estimations of inventory based on observation. Without an inventory, they will not be able to answer questions such as: *Which items are almost gone and need to be replaced? Which items were most popular? Which items should not be replaced because they did not sell well? How much profit did we make on each item?* This may cause the owners to overstock an item. Alternatively, they may not discover that they have run out of an item until a customer asks to purchase it. Then, the customer will have to wait for the company to order it and they may choose to purchase the item from a competitor that already has it in stock.

Another problem with the lack of inventory is that it leaves the company more susceptible to theft or misplacement of items. There will be no way to either detect or prove that items have been stolen since there is no record of how many items the company owns.

Significance and Scope

The problems discussed above arise from two main two main customer needs. The first need is for a more efficient mode of conducting transactions. The second is a need for a way to track inventory. Both of these can be resolved with the same solution: a computerized Point-of-

Sale (POS) System which can be used to facilitate transactions and be connected to a database to track incoming and outgoing items.

There are already several existing POS systems in use that have been demonstrated to be successful. The POS system for this project will be successful because the underlying infrastructure of the program will be similar to most other POS systems, leveraging their heritage and reliability.

Method

Plan Overview

The proposed application will consist of two main parts, the POS application and the database. The database will be connected to the POS so that it is updated each time a transaction occurs through the application portal. The system will be accessed online so that it can be run on any desktop device. It will be constructed on top of the pre-made infrastructure of a cloud-based Platform-as-a-Service (PaaS) which will manage the network, operating system, API's, middleware, and virtualization of the system as well as provide a web server and database. The developer will be responsible for building the front-end user interface, coding the back-end business logic, and setting up the logic of the provided database (see Figure 2). The selected programming languages to be used are CSS, HTML, and JavaScript for the front-end, and Python for the back-end. All these languages are compatible with the chosen PaaS, Microsoft Azure.

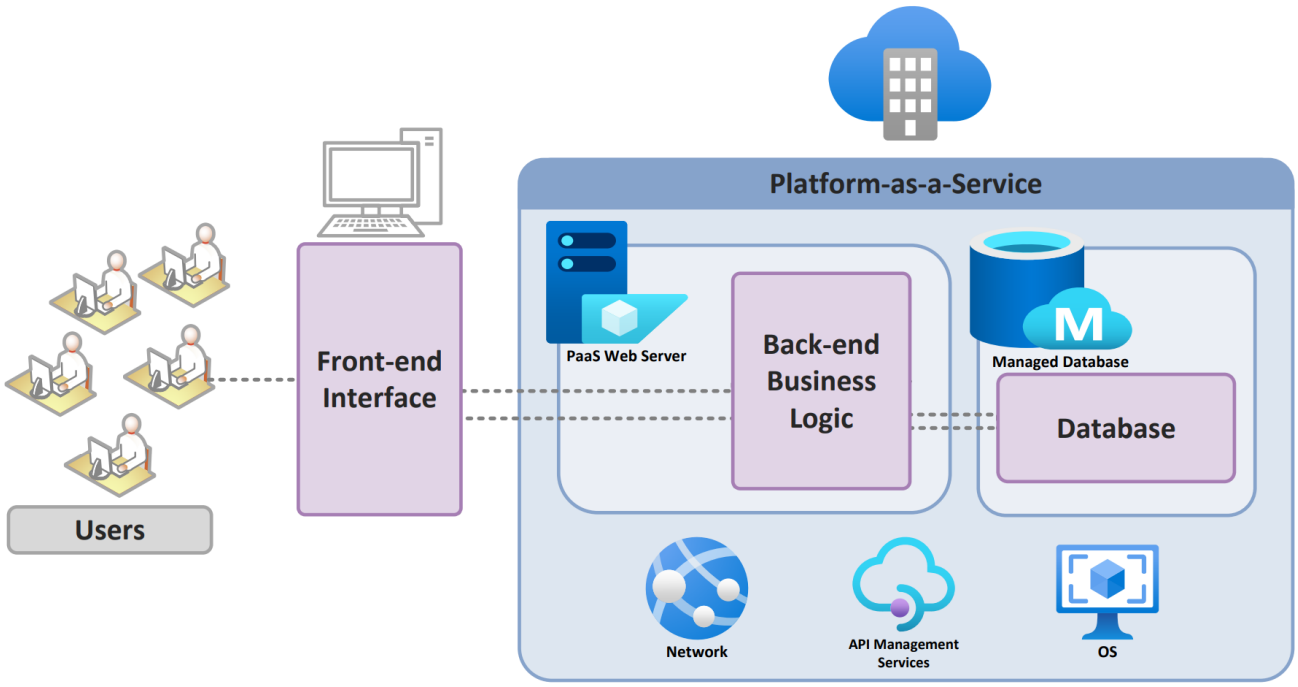

Figure 2: Application Block Diagram

The system will have a login portal and five main pages: “Home,” “Sale,” “Purchase,” “Customer,” and “Administrative.” The UML Activity Diagram in Figure 3 offers a simplified illustration of the application system logic. The “Home” page allows the user to navigate to any other page. The “Sale” and “Purchase” pages allow the user to search for and add/delete items that the client is buying or selling. The application automatically calculates the price for these. The “Customer” page allows the user to enter and save information about the customer. The user will be directed to this page any time a purchase is made. Finally, the “Administrative” page is restricted, only accessible to users of appropriate level. This is where changes to item price or quantity can be made, users can be added or deleted, and searches on transaction history can be made. Necessary requirements for the system are listed in Table 1 and stretch-goal requirements are listed in Table 2.

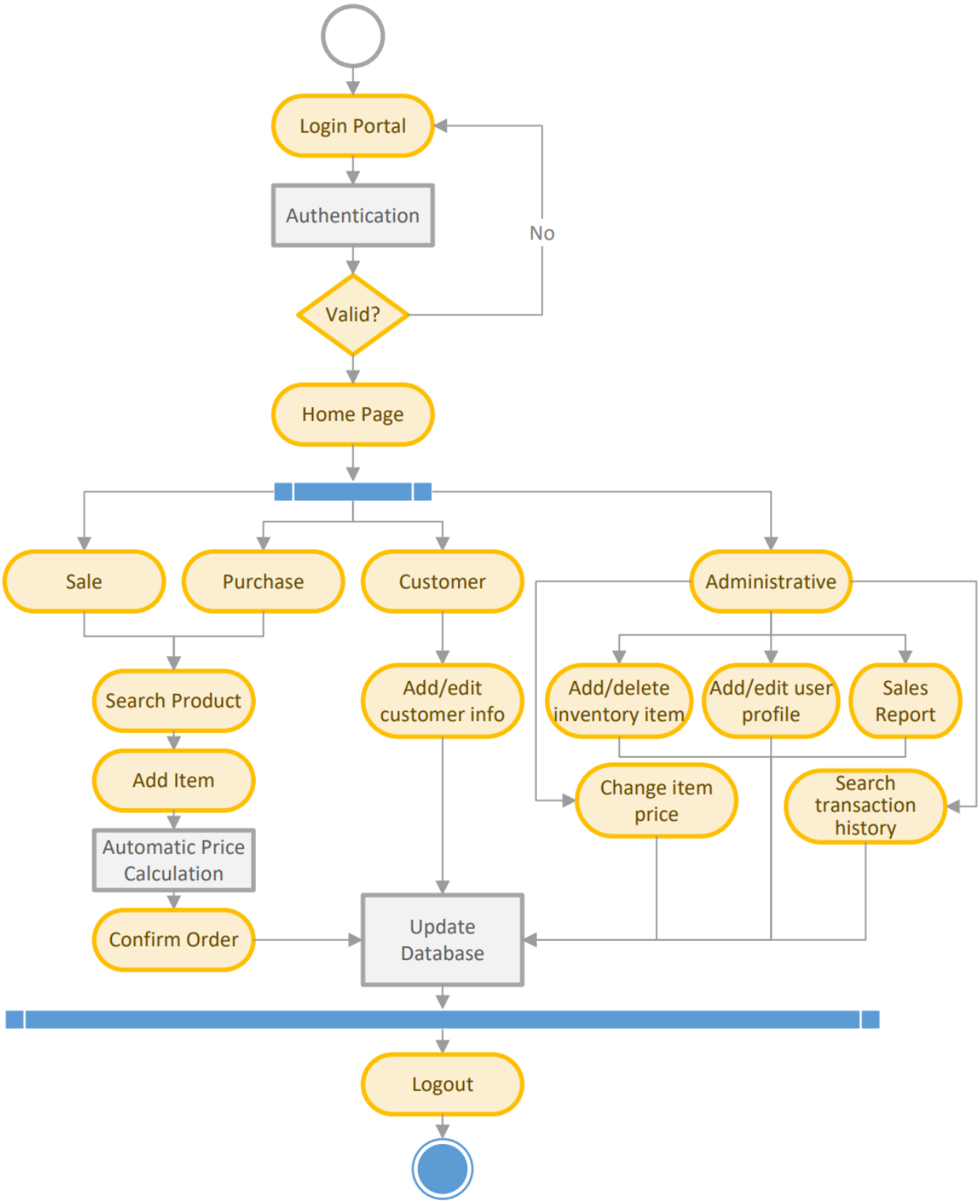

Figure 3: UML Activity Diagram

Table 1: Necessary Requirements

ID	Feature	Rationale
0.1	The system development and maintenance cost shall not exceed $55,000	The cost cannot exceed the customer's budget or the project will halt.
0.2	The development timeline shall not exceed 1 year	The customer must have their product in a reasonable amount of time.
SEC.1	The system shall require a login username and password to access the program	Since the system may have personal customer information on file (see SYS.5.1) as well as sensitive information about business stock, security must be in place to limit access.
SYS.1	The system shall perform purchase and sale transactions	The system should completely replace the current hand-written invoice system.
SYS.1.1	The system shall have a "Sale" page where the user can perform a sale by selecting multiple items and their quantity.	Necessary function to fulfill requirement SYS.1.
SYS.1.1.1	The "Sale" page shall contain buttons for the following actions: add item, remove item, change quantity, complete transaction, cancel transaction, payment type.	These are the most basic operations necessary for a sale transaction.
SYS.1.2	The system shall have a "Purchase" page where the user can perform a purchase by selecting multiple items and their quantity.	Necessary function to fulfill requirement SYS.1.
SYS.1.2.1	The "Purchase" page shall contain buttons for the following actions: add item, remove item, change quantity, complete transaction, cancel transaction.	These are the most basic operations necessary for a purchase transaction.
SYS.1.2.2	For a purchase, the system shall automatically redirect the user to the "Customer" page (see SYS.5) where the customer's info can be added before completion	The company is required to record customer ID for all purchases.

SYS.1.2.3	The system shall not allow the completion of a purchase transaction without customer info	This will ensure the user does not forget to input customer information.
SYS.1.3	The system shall automatically calculate prices based on user input	To reduce clerical error, the system should automatically perform price calculations.
SYS.1.3.1	The system shall connect to the current market value of precious metals (available online)	An accurate price must be calculated based on the current market value. Thus, the system must be able to access the market value.
SYS.1.3.2	The system shall connect to an editable form or spreadsheet indicating the company's premiums for each item	In addition to the market value, the price of each item is based on the company's premiums. This must be evaluated into the price calculation. It should also be editable (see SYS.6). Editing privileges can be limited to owners or managers (see SEC.1.2).
SYS.1.3.3	The system shall perform different price calculations based on whether a sale or purchase is being performed	The company buys items for less than they sell them. A common clerical error is buying an item for the sell price or vice versa. The system should recognize which action is being performed and calculate the price accordingly.
SYS.2	The system shall track company inventory in a SQL relational database	The company needs inventory information.
SYS.2.1	The system shall update the inventory database with every transaction	To achieve an updated inventory, the POS system should not only store the information, but consistently maintain it.
SYS.3	The system shall generate and save an invoice after transaction completion	The company must keep record of each transaction with an invoice.
SYS.4	The system shall enforce consistency in usage of terms	This requirement will avoid future confusion when reviewing transaction history. It will also ensure that the items selected by the user will match item identification in the inventory database and price spreadsheet so that they are recognizable and findable by the system. If the system did not recognize an item, it would not be able to calculate a price or update the inventory database.

SYS.4.1	Where possible, all prompts requiring user input shall force selection from a list of values (exceptions may be customer name or address)	This requirement achieves requirement SYS.4.
SYS.5	The system shall have a “Customer” page where the employee can input (or select – see SYS.5.1) customer info	The company is required to record customer ID info when purchasing items from them.
SYS.6	The system shall have an “Administrative” page where the user can add items or edit price premiums	This page will allow the user (can be restricted to owners – see SEC.1.2) to add a new item type to the inventory database or change the premium that an item is bought/sold for (see SYS.1.3.2).
SYS.7	The system shall have a “Home” page where users can navigate to other pages	Users will frequently need to revisit certain pages, not necessarily in sequential order, so they should be able to navigate to any page.
OPR.1	The system shall be an application usable on at least three devices simultaneously	The company needs to run the POS system on at least three different computers (one for the owner, one for the manager, and at least one for the employees to share). Ideally, the company would like to run the application on six computers simultaneously.

Table 2: Stretch-goal Requirements

ID	Feature	Rationale
SEC.1.2	The system shall create different user roles with varying levels of access	In addition to basic security, the employers may want to limit the employees’ ability to view certain sensitive information or their ability to makes changes to settings.
SYS.1.4	The system shall allow the user to override the automatic price calculation with a manual price input	The employee may want to give the customer a discount or pay extra for a certain item.
SYS.1.4.1	A second user’s approval shall be required for a price override	To ensure that no mistake is being made, if a user tries to override the automatic price calculation, a second user must input their credentials to approve the transaction. The approval could also be limited to owners/managers (see SEC.1.2).

SYS.4.2	Where it is not possible to implement a list of values, user entries shall be checked for type validity	Although an ID number, email, or phone number could not be implemented with a list of values, the system can check that the entry is valid by checking data type (number vs. character) and length (phone number should be 10 digits long including area code). This check will mitigate clerical error.
SYS.5.1	The system shall save and record customer information	In addition to inventory, customer info can be stored in the database. If the customer returns, their information can be automatically populated so that the employee does not need to input it manually.
SYS.5.1.1	The system shall not allow addition of an already existing customer	The system should not allow a user to add a customer with the same first name, last name, and address as a customer already in the database.
SYS.7	All transactions shall be searchable	The company may have need to search transaction history to view all within a certain time period, or all transactions involving a certain customer or item.
OPR.2	The system shall achieve reasonable platform independence	The company would like to use this system regardless of the operating system of the device it is running on.
NET.1	The system shall have the capability of communicating with a printer	The company would like to print their invoices to receive customer signature and to give the customer a copy.
REL.1	The system shall be available Mon-Sun (0800-1700) and downtime during these times shall not exceed 2 minutes in any one day	The system must be reliable that it will not malfunction significantly during business hours.
RSP.1	When a button is pressed on the system, the display shall be updated in less than 1.5 seconds.	The system must have reasonable responsivity so that it does not impede workflow.
RSP.1	If the screen takes longer than 1.5 seconds to update, a message shall display to indicate action progress	If the system takes longer than usual to respond, it should notify the user so that the user does not try to exit or resubmit the action.
USB.1	The system’s interface shall be adequately intuitive so that new users can be trained to use it within two days of training	To avoid disruption of workflow, the employees should be able to learn the new system somewhat quickly.

Project Timeline

The customer desires the product within 5-8 months. The estimated time required for the project is approximately 6.5 months. This allows a margin of 1.5 months for unexpected challenges and delays (see Figure 4).

The first two weeks will consist of meetings with the stakeholders to discover what the customer needs and define system requirements to meet those needs.

The next three months will consist of system design for the back-end, front-end, and database modules. These can be adapted from existing POS and database systems, so development can be expedited. The developer will design and plan the architecture for all modules simultaneously for about two weeks. When this is finished, the developer will build the back-end and database simultaneously, after which work on the front-end can begin.

The next month and a half will consist of integration between the three modules of the system. After integration, a pre-final exam will be conducted to ensure that the front-end, back-end, and database all work properly together.

Finally, the last two weeks will consist of final validation and documentation. The product will be configured, launched, and delivered to the customer. After launch, the developer will be available on an as-needed basis for maintenance.

ID#	TASK	START	END	DAYS	MARGIN
1	**Project Planning**	1/1/24	1/19/24	19	8
1.1	Conduct interviews with stakeholders	1/1/24	1/3/24	3	
1.2	Write User Stories	1/2/24	1/4/24	3	
1.3	Define System Requirements	1/4/24	1/10/24	7	
1.4	Schedule Margin	1/10/24	1/17/24	8	
1.5	◆ Requirements Document Completed	1/17/24	1/19/24	3	
2	**Create Back-end Features**	1/15/24	3/15/24	61	9
2.1	Identify heritage POS system	1/15/24	1/19/24	5	
2.2	Design & document adaptions for POS	1/18/24	1/24/24	7	
2.3	Develop code	1/24/24	2/23/24	31	
2.4	Test & revise back-end code functionality	2/26/24	3/5/24	9	
2.5	Schedule Margin	3/5/24	3/13/24	9	
2.6	◆ Back-end Midterm Exam Complete	3/13/24	3/15/24	3	
3	**Create Inventory Database**	1/15/24	3/8/24	54	5
3.1	Analyze & define data dependencies	1/15/24	1/24/24	10	
3.2	Design & document database	1/24/24	2/16/24	24	
3.4	Test & revise database functionality	2/19/24	2/23/24	5	
3.5	Schedule Margin	2/26/24	3/1/24	5	
3.6	◆ Database Midterm Exam Complete	3/4/24	3/8/24	5	
4	**Create Front-end Interface**	1/15/24	5/7/24	114	8
4.1	Identify heritage POS system	1/15/24	1/19/24	5	
4.2	Design & document front-end interface	1/18/24	1/22/24	5	
4.3	Develop front-end code	3/4/24	3/29/24	26	
4.4	Feedback from stakeholders	4/1/24	4/3/24	3	
4.5	Integrate feedback	4/3/24	4/12/24	10	
4.6	Test & revise front-end functionality	4/15/24	4/24/24	10	
4.7	Schedule Margin	4/24/24	5/1/24	8	
4.8	◆ Front-end Midterm Exam Complete	5/1/24	5/7/24	7	
5	**Integration**	5/7/24	6/28/24	53	11
5.1	Integrate back-end and database modules	5/7/24	5/17/24	11	
5.2	Integrate back-end and front-end modules	5/15/24	5/29/24	15	
5.3	Test & revise integration functionality	5/29/24	6/14/24	17	
5.4	Schedule Margin	6/14/24	6/24/24	11	
5.5	◆ Integration Pre-Final Exam Complete	6/24/24	6/28/24	5	
6	**Final Testing**	7/1/24	7/22/24	22	8
6.1	Requirements validation	7/1/24	7/3/24	3	
6.2	Full system validation	7/3/24	7/10/24	8	
6.3	Documentation & Revision	7/1/24	7/12/24	12	
6.4	Schedule Margin	7/12/24	7/19/24	8	
6.5	◆ Final Product Delivery	7/22/24	7/22/24	1	
7	**Maintenance**	7/23/24			
7.1	Ongoing Maintenance	7/23/24	7/22/25	365	
7.2	◆ Milestone				

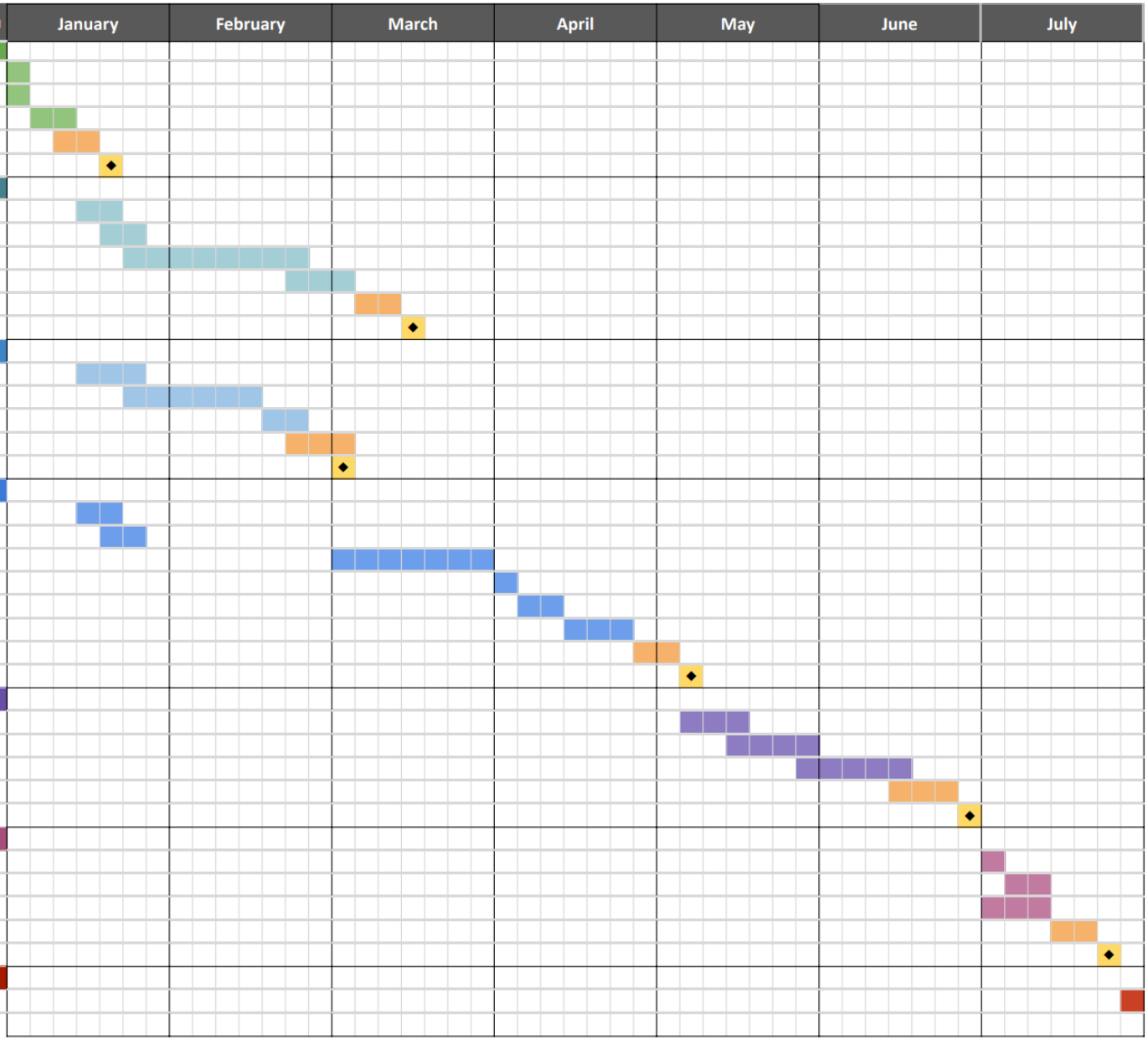

Figure 4: Project Timeline

Assessment and Testing

Ongoing unit tests will be performed throughout development to test small portions of code withing each module (front-end, back-end, database). In addition to these, the developer will perform mid-term and final exams.

Three mid-term exams should be performed on the component-level. The first mid-term should verify the back-end functional requirements and the logic of the program. For example, it will test that prices are calculated correctly and that authentication is required to log into the application. The second mid-term should verify the database logic. That is, it will test that the defined relations and schema are properly represented and items are given proper data types. The third mid-term should verify the front-end functional requirements of the user interface. For example, all pages should display the correct information in the proper format, each button should function properly, and the overall layout has no errors. These tests will include functional and active testing.

A system-level pre-final exam should be performed once the back-end, database, and front-end modules are fully integrated and the system is nearly ready to be launched. This exam will verify that all features are still functioning properly and that integration did not break any previously working code. It will verify that transactions correctly update the inventory and users with proper privileges can search transaction history. This exam can take a hybrid form with some regression testing that reuses tests from the previous mid-term exams, as well as functional testing, smoke testing, integration testing, endurance testing, and stress testing.

Once it has been established that the program functions properly, a final exam should be performed to validate the non-functional requirements of the system and ensure that it meets

customer needs. While the previous exams answer the question, “did we build the system right?” this final exam will answer the question “did we build the right system?” This exam will involve usability, reliability, availability, and vulnerability tests, as well as check general appearance.

Resources

Personnel

This project will be accomplished by a single software developer. Since this POS system will inherit many characteristics from existing systems and the infrastructure will be pre-established by the PaaS, it is not an unrealistic task for a full-stack developer.

Software Resources

Most software required for this project is publicly available at no charge (see Table 3). Github will be utilized to facilitate code development, version control, file storage, and informal documentation of system evolution. This documentation will be used later to create the software documentation and manual. Visual Studio Code is a common IDE and debugger used for code development and can be used for both coding and testing. Finally, since the system will be cloud-based, it will require a Cloud-based PaaS (Platform as a Service) on which to build the system and host the server.

Table 3: Software Resources

Resource	Purpose
Github	Version control; documentation of development evolution & issue resolution; file storage
Visual Studio Code	Development & debugging
Cloud-based PaaS & Database	A place to build and store the server, POS, and database

Hardware Resources

No hardware resources are required from the customer. The software developer will use their own device for development. The customer already owns desktop computers that the owners and employees use for research, communication, and other business operations. The application will be built to run on these devices.

Budget

The customer has a total budget of $50,000-$55,000. Approximately $35,000-$40,000 is allocated to initial development with $10,000-$15,000 for continued maintenance and services for the year following development.

The only resources that incur a cost are the developer and the PaaS. The cost to hire a software developer contractor varies, so an estimate of $85/hour was used to calculate the budget. The cost to pay the developer for initial development was based on the assumption that the developer has more than one contract and devotes to this project approximately four hours per day, four days per week, with four weeks per month. The cost of ongoing maintenance services was calculated based on 8 hours of maintenance per month at $85/hour. The Microsoft Azure PaaS is charged on an hourly subscription. Depending on the service chosen, the price varies widely. For this project, the A5 instance was chosen with 2 cores and 490 GB of temporary storage. After Long Beach, California sales tax [1] is added, the cost for this plan is $282 per month [2]. The total estimated cost for this project is $48,737 (see Table 4), leaving a $1,263-$6,263 margin for unexpected costs.

Table 4: Project Budget

Initial Cost			
Resource	Cost per hour	Cost per month	Total Cost (6.5 months)
Full-stack Developer	$85	$5,440*	$35,360
Microsoft Azure PaaS (pay as you go)	$0.35[2]	$282	$1,833
Total:			**$37,193**
Ongoing Post-development Cost			
Resource	Cost per month		Total Cost (1 year)
Maintenance (as needed)	$680		$8,160
Microsoft Azure PaaS (pay as you go)	$282		$3,384
Total:			**$11,544**
Total: $48,737			

*Estimate of 4hrs/day, 4 days/week, 4 weeks/month spent on this project.

Conclusion

As we have seen, a POS system is an appropriate solution to the needs of the customer, The Gold and Silver Exchange. The business will benefit from an accurate inventory; sales data for analytics which will help them make decisions about restocking & resale to help them optimize profits; an ability to search and review transaction history; and expedited transactions so that employees can spend less time on paperwork and more time on other productive tasks. Although The Gold and Silver Exchange is the main stakeholder in this case, the employees will also benefit from this new system as it will make their jobs easier by relieving paperwork. The customers will also benefit because their transactions can be conducted more quickly so their time waiting in the store will be shortened.

References

[1] (n.d.). *California city & county sales & use tax rates (effective July 1, 2023).* California Department of Tax and Fee Administration. Retrieved December 2, 2023, from https://www.cdtfa.ca.gov/taxes-and-fees/rates.aspx

[2] (n.d.). *Cloud services pricing*. Microsoft Azure. Retrieved November 25, 2023, from https://azure.microsoft.com/en-us/pricing/details/cloud-services/