Software Requirements Specification

for

E-commerce Loyalty Stack

Version 1.0 approved.

Prepared by Brad Davis, Tyler Parks, Sandy Martinez-Echegoyen,
Vispendra Chahar, Venkata Rama Reddy

Team Loyalty

02/20/2023

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Brad Davis, et al.	2/25/2023	Initial draft version for Deliverable 2	1.0

1. Introduction

1.1 Purpose

The product described in the document below is a Loyalty Point & Service Management System integrated with a mock E-commerce Storefront web application. As of this writing, the current document and project is in version 1.0. Our group's product will allow customers to shop for various items (*item types pending*) on a web storefront, then redeem and collect Loyalty Points based on purchases. This functionality requires several architectural layers including: the web store frontend, the web store backend, the Point Management System, customer account creation, and Point collection and redemption.

1.2 Document Conventions

This document will be described by sections of structured information reflected by:

- A Heading
  - Identified by the section number to the left of the heading.
- Content Paragraph(s)
  - One or more paragraphs of information describing the heading.
- Optional Diagram(s)
  - One or more diagrams supporting the text content.

Additionally, the list of our software requirements is broken up into 3 'phases' that we will work through over the course of the semester. Within each phase, there could be a hierarchy of requirements that take precedent over one another. For example, completing **Customer Account Creation** before beginning work on **Point Transactions**. These priorities will be described in the Development Phase Plan in section 6.

1.3 Intended Audience and Reading Suggestions

As this project is still in-development, the following document is intended for Team Loyalty Developers, Course Staff, and customer users and testers.

The following SRS Document is organized into chapters describing the software's overall description, system features, and functional and non-functional requirements. To understand what the system can accomplish, thoroughly read the full list of feature requirements in Chapter 3 and 4. To get a top-down understanding of how our group plans to meet these requirements, see Chapter 2; however, a full reading of this document will lead to a complete understanding of this software.

1.4 Project Scope

<u>Description and Purpose</u>
This project aims to produce a Loyal Service and Point Management System, integrated atop a mock website storefront. This model of business, seen used by airlines, the food industry, and banks, will not only persuade customers to reuse the service, but also convert external users to new customers. This is due to the model's ability to reward customers with each purchase made.

Benefits
Ranging from new customer acquisitions, to increased sales and site traffic, there are many benefits to creating software that employs this business model. Once users become new customers on our site, they will be immediately incentivized to return because of the instant reward they'll receive from a purchase.

Objectives
The main objectives of this software are listed below. Each of these objectives describes a high-level layer of the software's functionality.

1. E-commerce Storefront
2. Customer Account Creation
3. Shopping Cart and Payment Processing
4. Loyalty Management System

These layers will likely be completed in the order shown, or with some parallel overlap, due to the N-tiered structure our software was designed around.

Business Model
See the Project Proposal links below for the official 'scope document'.

1.5 References

Team Loyalty Project Proposal:
https://myunt.sharepoint.com/:w:/r/sites/CSCE5430TeamLoyalty/_layouts/15/Doc.aspx?sourcedoc=%7BF11EB40A-8DD7-4882-B881-A65EB6850C9F%7D&file=Team%20Loyalty%20-%20Project%20Proposal.docx&action=default&mobileredirect=true

Team Loyalty Project Proposal Slideshow:
https://myunt.sharepoint.com/:p:/r/sites/CSCE5430TeamLoyalty/_layouts/15/Doc.aspx?sourcedoc=%7BF35319FF-2608-4F46-9ABB-EF727457EA81%7D&file=Team%20Loyalty%20-%20Project%20Proposal%20Presentation.pptx&action=edit&mobileredirect=true

2. Overall Description

2.1 Product Perspective

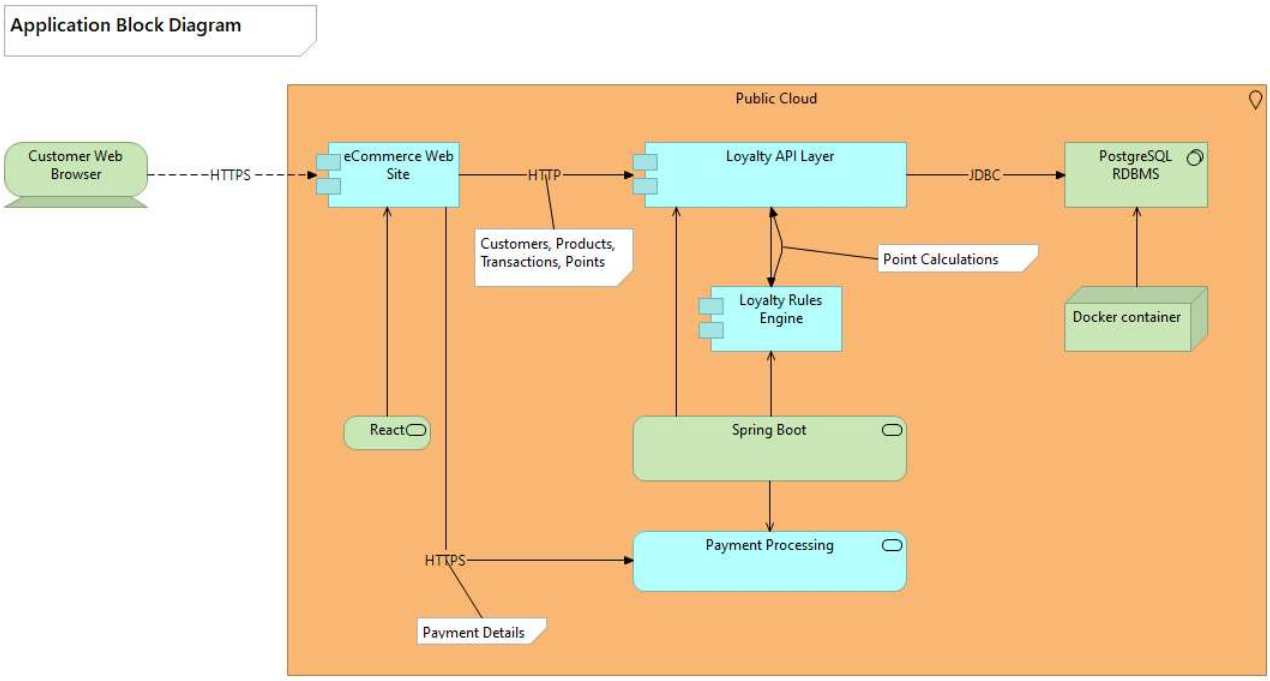
The E-commerce Loyalty Stack project represents a subset of a typical ecosystem of retail and e-commerce marketing technology systems. The primary focus of the project is an enterprise-style set of core loyalty capabilities. Loyalty solutions are often integrated after the fact into existing technologies like in-store point-of-sale solutions, e-commerce websites, mobile applications, email sending platforms, data warehouses, etc. To provide an integration point and facilitate the demonstration of key loyalty features, part of this project necessarily involves building a consumer-facing platform to apply those loyalty capabilities to. In this case a basic mock e-commerce experience will be part of the deliverable. However, the loyalty capabilities being delivered could then be integrated into other e-commerce solutions or some of the other types of systems previously mentioned.

There are many competing products and services in the loyalty commercial space, but few if any of them are available as free off-the-shelf OSS solutions. Upon reaching maturity, the core Loyalty

Stack deliverable could be offered as such a solution. However, as delivered this project will represent a Minimum Viable Product (MVP).

The system as implemented per these requirements will consist of a few major components. The e-commerce web site with loyalty enhancements will be integrated via service calls to a middle layer of loyalty APIs. The loyalty system will also host a rules engine for calculating promotional currency awards and access a backing data store. For demonstration purposes the project will also deliver a mock payment processing solution component.

The e-commerce web site will be accessible across public networks on the Internet for anyone to browse, shop or register using a web browser. The web site will access loyalty APIs, payment processing and any other support capabilities across private internal networks protected from public access. And the data store will be fully protected from external access and will be only available to the loyalty APIs and components via its native communications protocol.



2.2 Product Features

The E-commerce Loyalty Stack product will present a conventional web shopping experience to site visitors. Product information will be displayed and browsable by category. Visitors may register for a web site account to make purchases, and they will be given the option of joining the site's loyalty points program. Members of the program who submit purchase transactions to the web site will earn points for the amounts they spend and the products they purchase. In turn, program members will be able to redeem those points for discounts off future purchases at the website. The e-commerce experience as delivered will be a mock, and no payments will be charged, and no merchandise will be delivered. Site verbiage and controls may be put in place to mitigate any mistaken perceptions to the contrary. Operators of the E-commerce Loyalty Stack platform will be able to configure rules for promotional currency issuance for specific products, categories and spend amounts, and for specific date ranges.

2.3 User Classes and Characteristics

The primary end-users of the loyalty experience are simply web citizens from the general public who shop online. They are of average sophistication and should be able to easily navigate and use the shopping experience without much support. The core loyalty features should be easily recognizable to these users by promotion and emphasis within the e-commerce experience.

A second set of users would be the management and operations staff of the loyalty program. These are the people who are staffed by the e-commerce concern and are responsible for planning and configuring the promotional strategy of the loyalty program, including how many points customers can earn and for what activities and during what periods of time, as well as what customer loyalty points may be redeemed for.

2.4 Operating Environment

The product will operate in an Internet-facing web-centric environment. All application conventions and protocols will utilize web-based technologies. The product must be capable of being hosted on public cloud providers, and the underlying hardware will be virtualized as cloud compute resources. The compute environment is assumed to be industry-standard x86-64. The implementation languages (JavaScript, Java, SQL) and frameworks (React, Spring Boot) are primarily open-source, free-to-use technologies. Although these technologies are not particularly platform-specific, the product will initially be hosted on Linux-based operating systems and/or containers. Current stable LTS Ubuntu 22.04 or a close derivative will be targeted, and Node.js as well as Docker and related technologies may be leveraged. Product development and testing can generally be conducted on any major operating system or suitably modern desktop/laptop hardware but will be validated for the intended operating environment.

2.5 Design and Implementation Constraints

The e-commerce component of the product, consisting of the consumer-facing hosted web site application, is to be implemented in JavaScript and the React framework. Existing React-based shopping site components may be used as a development accelerator, but other components will need to be developed specifically for this product, especially those providing loyalty capabilities. The e-commerce module will be hosted as a node.js application (v18.14.2 LTS).

The back-end loyalty APIs, rules engine and any e-commerce mock components (e.g., payment processing) will be implemented in the Java programming language (JDK 17 LTS) and will use the Spring Boot framework (v3.0). APIs will conform to RESTful style and philosophy. These components will be hosted on the Spring Boot integrated Tomcat server.

The data store will be an RDBMS system, specifically PostgreSQL(v15.2). The database application will be container based for ease of deployment and implementation.

This system will not be deployed in a production-ready capacity and is expected to be able to function for demonstration purposes with relatively modest CPU and memory requirements, allowing utilization of minimum-sized virtualized compute instance types.

Th e-commerce site will communicate with customer web browsers via web standard HTTP/HTTPS protocols across public networks. The site will communicate with the API layer via the same HTTP protocols, and the API layer will communicate with the data store via its native DBMS protocol.

No issues are anticipated with availability of selected languages, tools, libraries, frameworks or operating systems or their amenability to implementation of the targeted capabilities. The main potential issue that could cause a revision of requirement is excessive scope if there is insufficient development time to implement all targeted features. In that case some of the essential features may be paired back as less critical or nice to have. Development should proceed in an iterative fashion with frequent re-evaluation of deliverable scope to fine-tune the final product capability set. A range of feature enhancements have been identified as potential stretch goals. These added capabilities are not included in the scope of the product at this time, but may be in future versions of this SRS as progress is evaluated.

2.6 User Documentation

Basic operational feature and usage documentation will be delivered in the form of a GitBook manual or documentation set.

Technical documentation artifacts may include architectural diagrams, Postman collections, Javadoc assets and/or Swagger.io API documentation [TBD].

2.7 Assumptions and Dependencies

The development of this product will be dependent on the availability of stable recent versions of several open source libraries and frameworks, including React for front-end and UI development, Spring Boot for API and middle tier development, and open source PostgreSQL RDBMS for data store capabilities. The team is also assuming the availability of one or more accelerator frameworks and datasets for e-commerce, product inventory, and payment processing. Operation of the product will also depend on Docker containers and cloud computer and storage services and their ability to host components implemented with these technologies.

3. System Features

3.1 E-commerce Storefront [Essential]

- 3.1.1. Web-based shopping site
  - 3.1.1.1. Banner graphic
  - 3.1.1.2. Color theme
  - 3.1.1.3. Mobile responsive design [Conditional]
- 3.1.2. Site visitors may browse products by category.
- 3.1.3. Site visitors may sort products by name or price.
- 3.1.4. Site visitors may search for products by name.
- 3.1.5. Product detail pages
  - 3.1.5.1. Display product name.
  - 3.1.5.2. Display product description.
  - 3.1.5.3. Display product category.
  - 3.1.5.4. Display product image.
  - 3.1.5.5. Display product SKU number.
  - 3.1.5.6. Display product price (in Dollars).
  - 3.1.5.7. Quantity entry and Add-to-Cart button.
- 3.1.6. Shopping cart
  - 3.1.6.1. Site visitor can click icon to display.
  - 3.1.6.2. Display items added to cart.

3.1.6.2.1. Display item quantity.
3.1.6.2.2. Display item name.
3.1.6.2.3. Display item (small) image.
3.1.6.2.4. Display item price (in Dollars).
3.1.6.3. Display calculated shopping cart subtotal.
3.1.6.4. Calculate sales taxes (Texas 8.25%)
3.1.6.5. Calculate shipping (flat rate)
3.1.6.6. Calculate purchase total.
3.1.6.7. Site visitor can change cart item quantity.
3.1.6.7.1. Update subtotal (in Dollars) when quantity changed.
3.1.6.7.2. Update sales taxes (Texas 8.25%)
3.1.6.7.3. Update purchase total.
3.1.6.8. Site visitor can remove item from cart.
3.1.6.8.1. Update subtotal (in Dollars) when item removed.
3.1.6.8.2. Update sales taxes (Texas 8.25%)
3.1.6.8.3. Update purchase total.
3.1.6.9. Proceed to Checkout
3.1.6.9.1. Site visitor shall be prompted to login if necessary.
3.1.6.9.2. Logged-in customer details are pre-populated:
3.1.6.9.2.1. Populate name.
3.1.6.9.2.2. Populate shipping address.
3.1.6.9.2.3. Populate email.
3.1.6.9.2.4. Populate phone.
3.1.6.9.3. Logged-in customer may update shipping details for order.
3.1.6.9.3.1. Update name
3.1.6.9.3.2. Update shipping address
3.1.6.9.3.3. Update email contact
3.1.6.9.3.4. Update phone
3.1.6.9.4. Logged-in customer can enter payment details.
3.1.6.9.4.1. Enter card type.
3.1.6.9.4.1.1. Multiple payment tender types supported.
3.1.6.9.4.2. Enter card number.
3.1.6.9.4.3. Enter card expiration.
3.1.6.9.4.4. Enter name on card.
3.1.6.9.4.5. Enter payment zip code.
3.1.6.9.5. Logged-in customer may submit order.
3.1.6.9.5.1. Validate shipping details complete.
3.1.6.9.5.2. Validate payment details complete.
3.1.6.9.5.3. Submit transaction details for processing.
3.1.6.9.5.4. Authorize transaction payment.
3.1.6.9.5.4.1. Return message on failure (e.g., Payment declined)
3.1.6.9.6. System will record authorized transaction to customer account.
3.1.6.9.6.1. Transaction Header data is recorded.
3.1.6.9.6.1.1. Transaction Date
3.1.6.9.6.1.2. Transaction Location (Web store)
3.1.6.9.6.1.3. Transaction Number (System generated)
3.1.6.9.6.1.4. Transaction Dollar Net Total
3.1.6.9.6.1.5. Transaction Dollar Gross Total
3.1.6.9.6.2. Transaction Line Detail data is recorded.
3.1.6.9.6.2.1. Line number
3.1.6.9.6.2.2. SKU
3.1.6.9.6.2.3. Quantity
3.1.6.9.6.2.4. Unit price
3.1.6.9.6.2.5. Net amount

3.1.6.9.6.2.6. Tax amount
3.1.6.9.6.2.7. Gross amount
3.1.6.9.6.2.8. Discount amount applied.
3.1.6.9.6.3. Transaction Tender data is recorded (single tender).
3.1.6.9.6.3.1. Payment type
3.1.6.9.6.3.2. Tender code
3.1.6.9.6.3.3. Payment details
3.1.6.9.6.3.4. Tender amount
3.1.6.9.6.4. Charge payment method.

3.1.7. Customer Profile
3.1.7.1. Site visitors can register and create a customer profile.
3.1.7.1.1. First and Last name (required fields)
3.1.7.1.2. Email address (required field, must be unique)
3.1.7.1.3. Secure password (required field)
3.1.7.1.4. Phone number (optional field)
3.1.7.1.5. Physical address – street, city, state, zip (optional, must be complete)
3.1.7.1.6. Birthday (optional field, month, and day)
3.1.7.1.7. Social login – Google, other [Conditional]
3.1.7.2. Customers can login with their registered email address and password.
3.1.7.3. Customers can log out.
3.1.7.4. Customer logged-in status is visible on the web site banner.
3.1.7.5. Logged-in customer can update their profile data.
3.1.7.5.1. Update first and last name.
3.1.7.5.2. Update email address (must be unique).
3.1.7.5.3. Change secure password.
3.1.7.5.4. Update phone number.
3.1.7.5.5. Update physical address.
3.1.7.5.6. Update birthday.
3.1.7.6. Customer can view purchase history.
3.1.7.6.1. Summary list of transactions including data, transaction number, total dollar amount, status.

3.2Loyalty Stack Point Management System [Essential]

3.2.1. Loyalty Enrollment.
3.2.1.1. Customer Profile enhancement - new and existing registered customers can join the E-commerce Loyalty program.
3.2.1.1.1. Checkbox on Customer Profile create/update page to join.
3.2.1.1.2. Accept T&C’s
3.2.1.1.3. System will capture customer’s active enrollment status and enrollment date.
3.2.1.1.4. System shall issue a unique loyalty card/account number to newly joined loyalty customer.
3.2.1.1.5. Customer account shall be initialized with a 0-point balance.
3.2.1.1.6. Customer will receive a Welcome bonus for joining loyalty [Optional]
3.2.2. Loyalty Points Earn
3.2.2.1. Loyalty Customers earn in a single currency: Points.
3.2.2.2. Points do not expire while the customer is an active member of the loyalty program.
3.2.2.3. Loyalty Customers can earn a program-defined Base points per dollar on eligible spend.
3.2.2.4. Base points are earned for net spend on eligible products, excluding taxes, shipping, or excluded products [rate TBD].

3.2.2.5. Bonus points can be earned on business-defined promotional basis for spend on specific products, product categories or spend threshold.
3.2.2.6. Bonus points can be additional points per dollar, fixed amount of bonus points, or point multipliers (2X, 3X, etc.).
3.2.2.7. Points will be calculated by a rules-based points engine.
3.2.2.7.1. Points can be earned on customer purchase transaction events.
3.2.2.7.1.1. Rules are entered and stored in the system and dynamically executed by the points engine when evaluating customer transactions.
3.2.2.7.1.1.1. Rule shall capture point eligibility criteria.
3.2.2.7.1.1.2. Rule shall capture point outcome calculation formula.
3.2.2.7.1.1.3. Rules shall be active for specific date ranges.
3.2.2.7.1.1.4. New rules can be entered into the system after it is live.
3.2.2.7.1.2. Authorized transactions for loyalty customers are evaluated in real-time by the points engine and point outcomes are issued to customer account immediately.
3.2.2.7.2. Points may be issued upon customer profile creation/registration in Loyalty program [Optional]
3.2.3. Loyalty Redemption
3.2.3.1. Shopping cart enhancement – Loyalty customers' available points balance is displayed during checkout.
3.2.3.2. Points redeemable for a single reward type: Purchase Discount.
3.2.3.3. Loyalty customer must have a minimum threshold points balance to redeem [TBD]
3.2.3.4. Points convert to dollars discount at a system-defined rate [TBD]
3.2.3.5. Shopping cart enhancement – available points can be applied to basket at checkout as $X dollars off.
3.2.3.5.1. Discount is prorated and applied to net amount of individual basket items.
3.2.3.6. Customer's earned points are redeemed and converted to discount rewards in FIFO sequence (oldest points first).
3.2.3.7. All redemptions are final, no points returned to balance, no residual cash value.
3.2.3.8. The conversion of points amount to reward of dollar amount is recorded in the system.
3.2.3.9. Transaction processing enhancement – Points redemption reward code/number and dollar amount recorded with transaction.
3.2.4. Loyalty Status
3.2.4.1. E-commerce web site enhancement – Logged in Loyalty customers will see Loyalty status indicator and loyalty card number on web site banner.
3.2.4.2. E-commerce web site enhancement - Loyalty status page/dashboard.
3.2.4.2.1. Display Loyalty status.
3.2.4.2.2. Display Loyalty card/account number.
3.2.4.2.3. Display points balance.
3.2.4.2.4. Display recent points history.
3.2.4.2.4.1. Display points earned records, including points amount, date and points bonus rule code/description.
3.2.4.2.4.2. Display point redemption records, including points amount, date, transaction number.
3.2.4.3. E-commerce web site enhancement – Loyalty transaction history
3.2.4.3.1. Loyalty customer transactions will display points earned for each transaction.

3.3Loyalty Program Admin UI [Conditional]

- 3.3.1. Integrated or separate UI for Loyalty program administrative access
- 3.3.2. Restricted to authorized privileged users.
- 3.3.3. Program Admin Capability
  - 3.3.3.1. Points engine rules entry from
    - 3.3.3.1.1. Capture eligibility criteria
    - 3.3.3.1.2. Capture calculation formula
    - 3.3.3.1.3. Capture rule date range
    - 3.3.3.1.4. Validate correct inputs.
    - 3.3.3.1.5. Manage rule status (Active/Inactive)
- 3.3.4. Customer Care Capability
  - 3.3.4.1. Search customer profiles by name
  - 3.3.4.2. Search customer profiles by account number
  - 3.3.4.3. Search customer profiles by email address
  - 3.3.4.4. View customer profile information
  - 3.3.4.5. View loyalty customer card/account number
  - 3.3.4.6. View loyalty customer points balance
  - 3.3.4.7. View customer purchase transaction summary history
  - 3.3.4.8. View customer purchase transaction details
  - 3.3.4.9. View loyalty customer points history (earn and redeem).
  - 3.3.4.10. Issue a manual point award to a Loyalty customer account.
    - 3.3.4.10.1. Manual point adjustment reason code, notes and positive/negative point amount will be recorded in the system and applied to the account balance.
  - 3.3.4.11. Inactivate loyalty enrollment upon customer request.
    - 3.3.4.11.1. Point balance will be reset to 0 upon enrollment inactivation.

4. External Interface Requirements

4.1 User Interfaces

The e-commerce web site is an essential deliverable for the product which will provide the primary product user interface. UI layout and design is still TBD, but general web conventions for clarity and ease of navigation should be followed, including simple menus, links, hierarchical or categorized navigational aids, and clear and obvious end-use controls such as form input fields and buttons with clear actions. Branding, theming, and color scheme should be tasteful and promote visual clarity. Any validation or error conditions should be handled and communicated gracefully to the user, and internal system issues, error codes, stack traces, etc. should not leak into the user presentation layer.

The conditional requirement for an administrative UI interface to manage the loyalty program can be more utilitarian. UI design should be basic but sensible, following standard form input norms and conventions. Any final layout and design are still TBD.

4.2 Hardware Interfaces

Although a virtualized x86-64 hardware operating environment is assumed, this product is largely hardware-agnostic and will rely on its hosting services, frameworks and libraries to interface as needed with underlying hardware resources and network interfaces.

4.3 Software Interfaces

The e-commerce web site, the middle layer loyalty APIs, payment processing and rules engine will all be operated via their respective hosting processes on the Ubuntu operating system (v22.04 LTS). The loyalty APIs will connect to an instance of PostgreSQL RDBMS software (v15.2).

The e-commerce web site will process general web browser requests, including page GETs and form POSTs. The e-commerce web site will pass messages to the loyalty APIs as a series of RESTful calls, using various HTTP verbs such as GET, POST, PUT, DELETE, etc. as appropriate. Messages passed will concern data about customers, products, purchase transactions, and loyalty points. The API layer will retrieve and persist these data entities into the data layer by a series of SQL CRUD operations native to the RDBMS.

4.4 Communications Interfaces

The e-commerce web site will need to support user access via web browsers in common use, including current versions of Chrome, Safari and Firefox. It will be hosted within a running instance of Node.js capable of handling HTTP requests. TLS support may be provided via an application load balancer.

The middle layer hosting loyalty APIs, payment processing and the rules engine will also support an HTTP interface. Web browser support is not targeted, and access must be authenticated and authorized from calling clients and applications. These application components will be hosted by the integrated Tomcat server in Spring Boot framework, and will respond to HTTP requests from client code and common API tools like Postman.

The data store will be a containerized instance of the PostgreSQL RDBMS system (via Docker containers). This should minimize deployment and configuration tasks for a relatively complex piece of software. Once the container is running it will be accessible from the middle layer using a JDBC driver (pgJDBC v42.5.4) on its standard TCP port 5432.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

Because this product is an educational project as part of a course of study, it is not anticipated to be deployed in a live production capacity at this stage. Performance requirements are not particularly stringent. For API requests there is target response time of 3 seconds or less for 95% of all calls. For web page interactions there is a target response time of 5 seconds or less for 95% of UI tasks. RDBMS queries will have a target response time of 500 milliseconds for 95% of all requests. Adequate logging should support these metrics. There will be a soft goal of the product beating these targets by substantial margins for demonstration purposes. The product should be able to maintain reasonable uptime for demonstration purposes and should be free of critical bugs or misconfigurations that cause it to cease functioning randomly or regularly.

5.2 Safety Requirements

This is not a mission critical system, and should not be capable of putting any process, operation, concern, property, or lives at risk of loss or damage. The system is also not meant to capture real customer data at this time. A good faith effort should be made in following sound implementation practices to produce a product that is not prone to data loss or infiltration, but the product is not to be considered fully security hardened at this phase.

5.3 Security Requirements

The e-commerce web site component should be capable of receiving and transmitting traffic over secure HTTPS protocol. Customer login functionality should be reasonably secure, and unauthorized users shall not be able to access a customer's profile and data.

The user-facing components shall implement OWASP Top 10 mitigations where applicable (https://owasp.org/www-project-top-ten/). The product should be resistant to basic data exfiltration attacks, SQL injection, XSS, etc.

The API layer and related components (points calculation engine, payment processing) shall be secured from public access and only be available for authenticated access by the user interface components.

The product is not envisioned for PCI scope at this phase. Payment processing is mock form only and should only deal with test payment card information. Real payment card information should not be entered, stored, or logged by the system.

5.4 Software Quality Attributes

All inputs to user interfaces should be validated for correctness, completeness, and formatting. Data submitted to APIs should be additionally validated and not trusted. System events and API requests and responses should be logged for detailed analysis and performance monitoring and should support multiple levels of detail. Conventions of good design in code construction will be followed and periodically evaluated to produce code artifacts that are maintainable and clear. Interfaces of all components will adhere to documented designs. Test coverage of all critical sections of code and as much additional code as possible will be targeted. The overall goal of the core loyalty capability components is that they can be redeployed and re-used, with or without enhancements in another operating context.

6. Other Requirements

6.1 Development Phase Plan

Implementation of the product will be delivered through 3 Project Phases:

Phase 1:
- Due Date: March 20, 2023
- Requirements: 3.1 (All)
- Functionality:
  - E-commerce web site supporting customer registration and login, product browsing, shopping cart, mock payment processing, and posting sales transactions to customer account. APIs and database entities to support these capabilities as outlined in the requirements.

Phase 2:
- Due Date: April 10, 2023
- Requirements: 3.2.1, 3.2.2, (3.3 Optional)
- Functionality:
  - Customer loyalty enrollment, customer points earn for purchases, supporting loyalty APIs, rules engine and database entities to support capabilities as outline in the identified requirements. Conditional development of some loyalty program Admin UI capabilities based on project bandwidth.

Phase 3:
- Due Date: May 1, 2023
- Requirements: 3.2.3, 3.2.4, (3.3 Optional)
- Functionality:
  - Customer redemption of points for discounts, view customer loyalty status, points balance and transaction history on e-commerce site. Complete conditional development of loyalty program Admin UI capabilities depending on project bandwidth.

6.2Member Contribution Table

Team Member	**Contributions**
Brad Davis	Block Diagram, Section 3 Requirements, Sections 2, 4, 5
Sandy Martinez-Echegoyen	Section 3 Requirements, Development Phase Plan
Tyler Parks	Section 3 Requirements, Section 1
Vishpendra Chahar	Technical research, Meeting minutes, Review and editing
Rama Reddy Venkata	Technical research, Requirements review

Appendix A: Glossary

API	Application Programming Interface
Container	A container is a software module that packages code and all its dependencies into a single unit, so the application runs quickly and reliably from one computing environment to another, while also providing a level of isolation from the underlying host system.
CRUD	Create, Read, Update, Delete. The four basic operations of most persistent storage and database management systems.
Docker	Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.
E-commerce	The process of selling products and services on the Internet leverage Web browsers, Web sites, payment processing and related web technologies.
HTTP/S	Hypertext Transfer Protocol (Secured), the communication protocol of the Web and Web browsers.
JDBC	Java Database Connectivity is standard API that allows Java programs to access database management systems. Libraries that provide access to specific database systems are implemented as JDBC drivers.
MVP	Minimum Viable Product. A minimum viable product is a version of a product with only enough critical features to be usable by early customers who can then provide feedback for future product enhancements.
Node.js	An opensource JavaScript runtime environment capable of hosting a large variety of JS packages, libraries and application code.
OWASP	Open Worldwide Application Security Project, which publishes lists of the most critical application security vulnerabilities facing organization.
PCI	Payment Card Industry. Typically, in reference to PCI DSS, Data Security Standards for handling and processing payment card data.
PostgreSQL	A free and open-source relational database management system.
RDBMS	Relational Database Management System, the classical style of database systems based on a relational model.

React	A JavaScript library for building user interfaces.
RESTful	(or REST) Representation State Transfer. An architectural style and design philosophy for building Web service APIs using the architecture of the Web.
SKU	Stock Keeping Unit, a form of product numbering system.
Spring Boot	Spring Boot is an open-source Java framework used to quickly create micro services.
SQL	Structured Query Language, a dynamic language for interacting with relational databases.
T&C's	Terms and Conditions
TLS	Transport Layer Security. The mechanism by which HTTPS traffic is securely encrypted and transmitted across the Internet.
Tomcat	Apache Tomcat is a free and open-source Java web application server that implements Jakarta Servlet, Jakarta Expression Language, and WebSocket technologies. It provides a pure Java HTTP web server environment in which Java code can run.
Ubuntu	A popular distribution of the Linux operating system, a free and open-source Unix-like OS.
UI	User Interface. Here, a Graphical User Interface (GUI) implemented either as a desktop application, a Web site, or a mobile device application.