Communications of the Association for Information Systems

Volume 54 Paper 10

3-5-2024

Wind Riders of the Lost River Range: A Modular Project-Based Case for Software Development

Kevin Parker
Idaho State University, parkerkr@isu.edu

Communications of the Association for Information Systems

Education Article DOI: 10.17705/1CAIS.05410 ISSN: 1529-3181

Wind Riders of the Lost River Range: A Modular Project-Based Case for Software Development

Kevin R. Parker

College of Business
Idaho State University
parkerkr@isu.edu
0000-0003-0549-3687

Abstract:

The Wind Riders of the Lost River Range (WRLRR) is a modular project-based simulated case study. This case study is designed to accompany a hands-on collaborative semester project for a variety of software development courses. The case can be used in courses like Systems Analysis and Design, Database Design, Software Engineering, and Software Development. The case provides a detailed account of a situation that closely simulates a real-world problem. This challenging experiential learning opportunity reinforces course concepts through their application in a realistic scenario. WRLRR has been developed using the modular design of teaching cases approach. This approach makes it possible to swap modules in and out of the case. This reduces instructor workload by making it unnecessary to develop a new case from scratch for each course iteration. It also discourages students from recycling solutions from previous courses because the case scenario has been altered. Adding or removing modules makes it possible to sufficiently vary the case study to refine the problem being addressed. Macros are provided to automate, and thereby facilitate, the insertion of modules.

Keywords: Teaching Case, Software Development Case, Systems Analysis and Design Case, Database Case, Front-end Development Case, Capstone Project Case, Data Analytics Case, Modular Project-Based Case.

This manuscript underwent editorial review. It was received 06/06/2023 and was with the authors for five months for one revision. Matti Mäntymäki served as Associate Editor.

1 Part 1: Case Details

1.1 Introduction

Serenity. Solitude. Adventure. All these and more can be found in the Lost River Range of the Rocky Mountains. This is one of the most remote and primitive areas in the United States. The Lost River Range has an abundance of deep gorges, high mountains, swift rivers, and frigid lakes. Heaven's Gate, Seven Devil Mountains, and King Mountain are just a few Idaho locations that are ideal for wind sports. People are drawn to the Lost River Range to experience the adventure of a lifetime. Wind Riders of the Lost River Range (WRLRR) is a specialty sports shop in central Idaho. The business specializes in the sales of wind sports equipment, like hang gliders, paragliders, and snowkites.

WRLRR is doing well with both an expanding customer base and improved financial statements. Even so, the company is proactively seeking ways in which to improve. However, any modifications to sales or services require updates to their IT infrastructure, and that's where you come in.

This simulated project-based case casts students in the role of software development consultants. In a database course, students will play the role of designer and implementer of the all-important database. In analysis and design or software engineering courses, students will analyze user requirements and design a new system. In software development courses, students will develop a software system to support the business's entire range of sales and services. This may involve both front-end and back-end aspects. In a data analytics course, students will decide what historical data needs to be replicated in a data warehouse. In addition, they may determine which internal and external trends should be tracked and analyzed to strategically guide future decisions.

In summary, the key issue is the design and development of a software system to support the operations of WRLRR. The current IT infrastructure needs to be assessed and modified. Further, any future changes made by the company must be technologically supported.

The scenario described below provides details about the baseline case only. Your instructor will supplement the case with additional details, so you can expect more to come.

1.2 Business Background

WRLRR is a growing enterprise in central Idaho. The business sells wind-propelled sports equipment such as hang gliding, paragliding, and snowkiting equipment. The business was founded as a co-op in 1990 as Lost River Wind Riders (LRWR). The founders were a group of friends who shared a passion for wind-propelled sports. The co-op allowed owners and a few friends to buy top-tier equipment for hang gliding and paragliding at reduced rates. They observed more enthusiasts becoming involved in wind-propelled sports. This provided an opportunity to leverage their avocation into a full-time business while continuing to participate in wind-propelled sports.

The company was a huge success. However, over time the owners realized that operating the company required sacrificing opportunities to ride the winds. They wanted to spend more time in the air rather than in the office. In 2014, Nick Lovick, an Australian from the Illawarra, visited the Lost River area to experience Idaho hang gliding. Nick fell in love with the solitude and terrain. He decided to make his home in Idaho and approached the owners of LRWR about purchasing the business. The owners reluctantly decided to sell. In 2015, LRWR was purchased by Nick Lovick. Nick decided to rename his company Wind Riders of the Lost River Range. The new name serves two purposes. First, it differentiates the business from its predecessor. Second, it maintains continuity so existing customers can easily find the company. In 2021, Nick was financially able to begin considering changes in the sales and services provided by WRLRR. Nick is the key decision-maker, but he consults the previous owners and his family on all business decisions.

1.2.1 Equipment Sales

The company's primary focus is on the sale of hang gliding, paragliding, and snowkiting equipment. The market for wind sports equipment in the central Idaho area has been booming. Annual sales increased from $20,000 in 1990 to more than $840,000 in 2004, to almost $1,250,000 in 2023.

While the company once kept a large inventory on hand, WRLRR has considered newer strategies in recent years. They adopted a hybrid retail model that combines the traditional retail model with micro-retailing and showrooming. In the traditional retail model, stores display and stock a large selection of items. Micro-retailing refers to the practice of having only one of each item in stock and having it on display. When a customer makes a purchase from a micro-retailer, the store immediately orders the product. The product is delivered to the customer within a few days. A similar approach, showrooming, delivers personalized customer experiences. Retailers focus on high-touch in-person experiences and long-term customer loyalty rather than on stocking extensive inventory.

WRLRR's hybrid model is implemented by stocking multiple items of top-selling wind sports equipment in inventory. They keep only display models of other items. When customers select items that are not stocked, they pay for the item in advance. WRLRR places an order for that item within hours. Whether the item is stocked or not, customers initiate a purchase in the same way. They make their selection from the showroom floor and bring the item's product selection card to the checkout.

1.2.1.1 Hang gliders

Hang gliders (Figure 1) are non-motorized, large triangular wings made of light, tough fabric over an aluminum frame. The pilot is suspended horizontally below the wing in a harness. The harness allows the pilot to control the direction using only their body. Takeoff can be achieved by launching into the air from a cliff or hill. Hang gliders rely on gravity as their source of propulsion, always falling downward. However, by seeking thermals and other uprising air currents, a skilled pilot can remain aloft for hours.

Figure 1. Hang Glider Action Shots

1.2.1.2 Paragliders

Paragliders, as shown in Figure 2, are non-motorized, foot-launched, ellipse-shaped wings (or parafoils). They are flown and landed with no other energy requirements than the wind and gravity. Unlike hang gliders, there are no metallic frames. The canopy consists of two layers of rip-stop nylon sewn together to form a gap between the two. Vertical fabric ribs support the gap, forming cells between each rib. The leading edge of the wing allows air to enter the cells, inflating the canopy for gliding. The canopy is attached to a harness that has a seat into which the pilot is strapped before flight. The paraglider is controlled by pulling gently on the control lines. Paragliders soar more slowly than hang gliders, which makes it easier for people to learn to fly them.

Both hang gliders and paragliders soar on currents of air. Both can stay aloft for 3 hours or more, climb to elevations up to 15,000 feet, and travel vast distances. Hang gliders have a "cleaner" aerodynamic profile and generally are capable of flying at much higher speeds than paragliders. However, paragliders can fly with lower wind speeds, requiring only about 12.5 miles per hour winds to take off, while hang gliders require winds of at least 25 miles per hour to launch.

Figure 2. Paraglider Action Shots

1.2.1.3 Snowkites

Hang gliding and paragliding are most common from May through September. In the interest of generating steady income year-round, the company extended its sales line into winter sports. To maintain the theme of wind-propelled sports gear, they decided to add snowkiting equipment to their line of offerings.

Snowkiting, shown in Figure 3, has been referred to as the evolution of kitesurfing. It combines the airfoil and techniques used in kitesurfing with the footgear and gliding surface used in snowboarding. Large highly controllable foils or inflatable kites propel boarders across the snow with just the power of the wind. Snowkiters can even travel uphill with ease if the wind is blowing in the right direction. All that is needed is the wind, a snowkite, skis, or snowboard. The snowkite is attached to a body harness by kite wires. There is typically a control bar that allows the kite to be turned by pulling one line or the other. Snowkiting is possible in winds from 5-30 miles per hour.

Figure 3. Snowkiting Action Shots

A

Insert the corresponding module segment to provide an overview of additional sales items selected to supplement the existing system.

1.2.2 Services and/or Activities

B

Insert the corresponding module segment to provide an overview of the new service or activity selected to supplement the existing system.

1.2.3 Facilities

The present location has a showroom and a warehouse. The showroom is a forty-five hundred square foot expanse, allowing sufficient room to examine several available models. A nearby escarpment provides an ideal setting for customers who want to “test drive” the equipment. The company now employs approximately twelve people in various capacities including sales, management, general support, etc.

1.3 System Description

1.3.1 Listing of Customer-Facing Processes

The current system supports the core business requirements of the enterprise. WRLRR provides several customer-facing processes:

- Customer purchases item(s)
- Generate purchase agreement
- Customer returns item(s)

A1

Insert the corresponding module segment to *list* customer-facing processes associated with additional sales items selected to supplement the existing system.

B1

Insert the corresponding module segment to *list* customer-facing processes associated with the new service or activity selected to supplement the existing system.

1.3.2 Listing of Back-Office Processes

Additional back-office services support business requirements. All of the reports in the following list are sent to management and archived in a report archive file. (Note: It may be one archive file or one file for each report.)

- Process returned inventory
- Order new inventory
- Process new inventory
- Generate sales report
- Generate current inventory report
- Generate salesperson sales report

A2

Insert the corresponding module segment to *list* back-office processes associated with additional sales items selected to supplement the existing system.

B2

Insert the corresponding module segment to *list* back-office processes associated with the new service or activity selected to supplement the existing system.

1.3.3 Descriptions of Customer-Facing Processes

The WRLRR computer system makes use of barcodes to record information. Each item in inventory is assigned a unique item number for inventory control.

1.3.3.1 Customer Purchases Item(s)

A customer may purchase one or more items. A purchase follows the script below:

1. The customer makes a selection and then brings the item or the associated product card to the checkout. The UPC can be found on either the item or the product card
2. The employee selects "Sale" from the main menu.
3. The employee requests the customer’s phone number to determine if the customer is on the customer list.
    a. If the phone number is not found, then the Customer Entry Form is displayed. A customer number is generated and assigned to the customer. Finally, their personal information is entered and added to the customer file, including:
        - Name
        - Address (include city, state, zip)
        - Telephone
        - E-mail address

    b. If the phone number is found, it indicates that the customer is a returning customer. The customer is asked if they need to update their profile. If they do, the following items can be updated via the Customer Update Form.
        - Name
        - Address (include city, state, zip)
        - Telephone
        - E-mail address
4. Then, the Sale form is displayed. The Sale form includes the following fields:
    - Salesperson number (obtained when the salesperson initially logged into the system)
    - Customer number (obtained in the preceding step)
    - Payment type
5. The UPC on each product or placard is scanned and the local inventory database is queried for the item's price. That price should match the price the customer saw on the item’s display. For example, the database reflects that the UPC represents a Skywalk Cayenne 6 paraglider that costs $4600.00. The following fields on the Sale form are populated:
    - Item category type (hang glider, paraglider, etc.)
    - Item number
    - Description
    - Serial number (if available)
    - Price
6. When the transaction is completed, the salesperson selects the total option. The system computes any sales tax and displays the total amount due.
7. The customer provides a credit or debit card. Alternatively, the customer may pay with cash. If a card is used, the cardholder presents their card (swipe, tap, insert, or other secure method). Credit/debit cards have a chip containing an ID. This ID is read by a near-field communication (NFC)-enabled point-of-sale (POS) card reader.

    Once the card information is acquired, the authorization process begins. Authorization is the process in which the card issuer approves or declines a transaction. Merchants use it to ensure customers have sufficient funds available to pay for a purchase.

    The credit card information enters a payment gateway. The payment gateway serves as a connector between the merchant and a payment processor. The payment processor sends this information to the card brand (such as Visa or Mastercard). The card brand then sends this information to the card issuer. The issuer validates the card number and verifies that the credit line has sufficient funds to approve the transaction. The issuer also checks the card verification value and ensures the billing address matches what is on file.

If the transaction is approved, the merchant receives authorization. The card reader or POS terminal displays a message that says, “Approved.” If a credit or debit card is used then the following fields on the Sale form are populated:

- Card type (if applicable)
- Card number (if applicable)
- Expiration date (if applicable)
- CVV (if applicable)

8. The employee saves a digital copy of the Sale form. Two copies of the purchase agreement are printed, which the customer signs. One copy is retained in a paper file and one copy is given to the customer as a receipt. The purchase agreement is detailed below.
9. If the item is in inventory, it is retrieved and given to the customer. The inventory is decremented to reflect the sale of each item. Otherwise, an order is submitted for the item.
10. The customer is also given their credit/debit card receipt if payment was by card.

1.3.3.2 Generate Purchase Agreement

A purchase agreement must be generated for every sale. As stated earlier, the employee prints two copies of the purchase agreement, which the customer signs. One copy is retained in a paper file and one copy is given to the customer as a receipt. The purchase agreement contains the following information:

- Header Section: Current date/time, salesperson number, customer name, customer address, customer city/state/zip, customer phone, customer e-mail
- Detail Section (for each item): Item number, description, serial number (if available), sales price
- Footer Section: Sales subtotal, sales tax (see business rules), grand total, payment method, masked credit card number

1.3.3.3 Customer Returns Item(s)

When a customer returns an item, the following events take place:

1. The customer indicates the need to return an item.
2. Returns require proof of purchase.
   a. If the customer provides proof of purchase (receipt or purchase agreement) along with the return item, the return process proceeds.
   b. If no proof of purchase can be provided, then the system can be queried for proof of purchase.
      - This is done by entering the customer number, customer phone number, or customer name/date of purchase.
      - If proof of purchase cannot be confirmed, the return request is declined.
      - If proof of purchase is confirmed the return process proceeds.
3. Returns must take place within the 90-day period noted in the business rules.
   a. Once proof of purchase has been confirmed, its date is checked to determine if the return falls within the return period.
   b. If the item is returned beyond the return period, no refund can be issued. Instead, the customer will be advised to contact the manufacturer.
   c. If the item is returned within the return period, the condition of the item must be assessed.

4. The employee next checks the item to be sure that the item has not been damaged by the customer.
    a. If there is damage to the item, no refund will be issued.
    b. If there is no damage to the item, a refund will be issued.
5. Transaction data is recorded per the following:
    a. The employee selects "Return" from the main menu. The Return form contains the following items:
        - Transaction date
        - Employee number
        - Customer number
        - Customer reason for return
        - Item condition per employee evaluation
        - Item number
        - Description
        - Serial number (if available)
        - Price
        - Refund decision
        - Designation (return to inventory or sell to discount retailer)
    b. The date returned is provided by the system. The employee number is provided when the employee logs in. The customer number, item number, description, serial number (if available), and price are obtained by scanning the receipt. They can also be retrieved from the proof of purchase. The employee selects the reason for the return on the form.
    c. The return decision is recapped below:
        - If proof of purchase cannot be confirmed, the item cannot be returned.
        - If the item is returned beyond the return period, no refund can be issued. Instead, the customer will be advised to contact the manufacturer.
        - If the item is returned within the return period but has been damaged, no refund will be issued.
        - If the item is returned within the return period undamaged, a refund will be issued.
    d. The refund decision is recorded.
    e. If appropriate, the customer is provided with a refund.
        - The system will default to using the original payment process. This means that the refund will be made using the same method by which the payment was made.
            - If the purchase was made in cash, a cash refund is provided.
            - If the purchase was made by card, the refund is applied to the card used to make the purchase. The payment processor receives the refund request and then sends the information to the card issuer. If the card issuer approves the refund, WRLRR generates a chargeback to return that amount to the card issuer. Finally. the card issuer posts a credit to the cardholder’s account.
    f. If the product was not defective, then it is flagged to be returned to inventory. If it is defective, then the item will be flagged to be sent to a discount retailer.
    g. A digital copy of the return form is saved.
    h. The customer will receive a receipt indicating the refunded amount and the refund type (cash, credit card, debit card).

A3 Insert the corresponding module segment to *describe* customer-facing processes associated with additional sales items selected to supplement the existing system.

B3 Insert the corresponding module segment to *describe* customer-facing processes associated with the new service or activity selected to supplement the existing system.

1.3.4 Descriptions of Back-Office Processes

1.3.4.1 Process Returned Inventory

Any returned items must be processed:

1. Some returned items are returned to inventory, per their designation attribute.

   The UPC is scanned along with the item number. Next, the item record is moved from the sales file back into inventory. Finally, the inventory count for that item type is incremented.

   a. The item is taken to the warehouse and stored in its proper location.

2. Other returned items are sent to a discount retailer, per their designation attribute.

   a. The UPC is scanned along with the item number.

   b. The discount retailer is contacted to arrange shipment of returned items. The original cost of the item is included.

   c. The discount retailer receives the returned item and sends reimbursement for the item. The reimbursement amount is typically around 75% of the original cost.

   d. To learn more, see https://www.netsuite.com/portal/business-benchmark-brainyard/industries/articles/wholesale-distribution/returns.shtml

1.3.4.2 Order New Inventory

1. An order for new inventory is generated under the following circumstances:

   a. A customer purchases an item that is not in stock. WRLRR must then place an order for that item within hours.

   b. An employee runs a periodic inventory report (explained below) that indicates the total number of items in stock.

      i. If the number falls below a preset limit, a reorder request is generated. The number of items ordered will equal the limit minus the current number. Preset limits are based on projected demand for an item type and are determined outside the scope of this system.

      ii. Before an order is placed, the reorder quantity is compared to the projected demand for an item type. The remaining sales season is taken into account, and order quantities can be adjusted.

   c. The price for each item type is obtained from the supplier (outside the scope of this system). That price is then entered into the purchase order. Shipping charges are also provided by the supplier. The tax rate is obtained from local governmental tax regulations. The subtotal, total tax, and total are all calculated.

   d. If the order is for a non-stocked item, the customer's shipping address will be included.

2. Orders for the same supplier are combined into a single purchase order.

3. Purchase orders are sent via email to suppliers who have provided an email address for their sales representative. Otherwise, they can be faxed or mailed to the appropriate supplier.

1.3.4.3 Process New Inventory

Assume that all orders placed to a given supplier are received in full. Further, assume there is no need to make an allowance for partial shipments and backorders.

1. When a shipment is received, an employee enters the following information into the system for each item:
    - Serial number (if available)
    - Item number
    - Item category (selected from a product category list)
    - Cost
    - Purchase date
2. The system creates a new record in the inventory table for each item. Then it initializes additional fields as follows:
    - Item number is automatically generated by the system in sequential order and is used as an inventory tag.

1.3.4.4 Generate Sales Report

The sales report can be generated daily or periodically, depending on business rules. As noted above, the report is sent to management and archived in a report archive file. It includes a listing of all sales for a specified period, including:

- Header Section: Begin date, end date
- Detail Section: Category number, description, quantity sold, sale price, total sales
- Footer Section: Grand total

1.3.4.5 Generate Current Inventory List

Periodically an employee may find it necessary to generate an inventory report. Such reports are used when it is necessary to take inventory of current stock, and when restocking inventory. The report includes the following:

- Header Section: Current date/time
- Detail Section (for each item): Item number, description, count, reorder quantity, serial number (if available), retail cost, purchase date

1.3.4.6 Generate Salesperson Sales Report

The salesperson sales report can be generated for one or all salespersons as needed, depending on business rules. As noted above, the report is sent to management and archived in a report archive file. It includes a listing of all sales data, including:

- Header Section: Begin date, end date
- Detail Section: Salesperson number, sales count, sales amount
- Footer Section: Total count, total sales

A4 Insert the corresponding module segment to *describe* back-office processes associated with additional sales items selected to supplement the existing system.

B4 Insert the corresponding module segment to *describe* back-office processes associated with the new service or activity selected to supplement the existing system.

1.3.5 Drivers of System Modifications

The software that supports the day-to-day activities was developed by an IT consulting firm several years ago. That software was sufficient to support the business's traditional retail model that was in place when Nick purchased WRLRR.

However, the company has transitioned to a form of “clicks and mortar” hybrid retail model. This hybrid model combines a traditional retail model with micro-retailing. The hybrid model, along with evolving customer needs, has made it necessary to improve information access and recordkeeping. Several business objectives initiated the change process:

- As their customer base and sales have increased, WRLRR has begun to experience problems with inaccurate inventory counts. In addition. inaccurate UPC data that has resulted in disgruntled customers, raising concerns about customer retention and lost sales.
- In addition, the customer return process can be challenging. Employees sometimes experience difficulty locating the correct sale.
- The company has been unable to determine sales history and trends from the current system.
- The success of their hybrid retail model can be improved by making enhancements to their system. They need a feature that can ensure that needed items are purchased and less popular items are not.
- Further, the business seems to be stagnating. The financial numbers are looking fine, but the company no longer feels as adventurous or edgy. WRLRR must attract not only new customers but a wider demographic of customers including thrill seekers. Adrenaline junkies often pursue even more extreme sports, and WRLRR can provide what they seek.

C

Insert the corresponding module segment to provide an overview of problems encountered in the existing system.

D

Insert the corresponding module segment to provide an overview of strategies under consideration for the future system.

1.3.6 Strategies Adopted for Future System

E

Insert the corresponding module segment to provide an overview of a *new* sales item selected to added by the future system in accordance with strategies adopted.

F

Insert the corresponding module segment to provide an overview of a *new* service or activity selected to be offered by the future system in accordance with strategies adopted.

1.3.7 Future System Description

1.3.7.1 Listing of Requested Customer-Facing Processes

E1

Insert the corresponding module segment to *list* customer-facing processes associated with new sales items selected to added by the future system.

F1

Insert the corresponding module segment to *list* customer-facing processes associated with a new service or activity selected to be offered by the future system.

1.3.7.2 Listing of Requested Back-Office Processes

E2

Insert the corresponding module segment to *list* back-office processes associated with new sales items selected to added by the future system.

F2

Insert the corresponding module segment to *list* back-office processes associated with a new service or activity selected to be offered by the future system.

1.3.7.3 Descriptions of Requested Customer-Facing Processes

E3

Insert the corresponding module segment to *describe* customer-facing processes associated with new sales items selected to added by the future system.

F3 Insert the corresponding module segment to *describe* customer-facing processes associated with a new service or activity selected to be offered by the future system.

1.3.7.4 Descriptions of Requested Back-Office Processes

E4 Insert the corresponding module segment to *describe* back-office processes associated with new sales items selected to added by the future system.

F4 Insert the corresponding module segment to *describe* back-office processes associated with a new service or activity selected to be offered by the future system.

1.3.8 Business Rules

1.3.8.1 Existing Business Rules

A partial list of business rules associated with WRLRR includes the following:

- A general list of product categories includes:
    - hang glider
    - paraglider
    - snowkite
- Each inventory category can have multiples of each item in stock, or none at all.
- When a purchased item is not in stock, an order for that item must be placed within 3 hours.
- Each specific inventory item, even those from the same inventory category, can have a unique price. These prices differ due to certain circumstances, such as price reductions, etc.
- WRLRR does not sell parts directly.
- The return period for purchases is ninety days.
- Each non-return transaction will be charged a 7.5% sales tax.
- The Payment Card Industry Data Security Standard (PCI DSS) guides organizations that handle branded credit cards from major card schemes. PCI DSS does not prohibit the collection of card verification codes/values prior to authorization of a specific transaction. However, card verification codes/values cannot be retained after the authorization of the transaction for which it was collected. Anyone who accepts credit cards in the United States agrees to comply with PCI DSS. While not a law, this industry standard is agreed to when a business contracts to accept credit cards. If a business accepts credit cards, it agrees to comply with PCI as part of the registration process. Some states have laws that support, to various degrees, PCI as a security and privacy standard.
- An employee is assigned an employee number, which may be referred to as a salesperson number for salespersons. Designating employee numbers occurs beyond the scope of this system. Assume employees have one and use it when necessary.

A5 Insert the corresponding module segment to list new business rules associated with additional sales items selected to supplement the existing system.

B5 Insert the corresponding module segment to list new business rules associated with the new service or activity selected to supplement the existing system.

1.3.8.2 New Business Rules

E5 Insert the corresponding module segment to list new business rules associated with new sales items selected to added by the future system.

F5 Insert the corresponding module segment to list new business rules associated with a new service or activity selected to be offered by the future system.

1.3.9 Overlooked Features

Part of a software developer's responsibility is to recognize if anything has been overlooked. For example, when a customer makes a purchase, were any necessary associated processes left out? This is where your familiarity with the normal functioning of businesses can come in useful. If you suspect something was overlooked, diplomatically bring it to the attention of the individual(s) who commissioned this project.

1.4 Overall Project Objective

WRLRR is not the small business it once was. It has become a staple of central Idaho and has the customer base to match it. When a business outgrows its capabilities, the solution is improving, upgrading, and innovating. As the market for wind-powered sports equipment becomes more competitive, a business must improve and evolve to be successful

It is vitally important for businesses to scale their software solutions as they grow. WRLRR has grown in the past three decades. As a result, their software infrastructure must address both their current and future needs. The wind sports equipment industry is far removed from the technology sector. Despite that, WRLRR must adapt as new technology becomes available. WRLRR is poised for rapid growth. However, growth is impeded by a system that is unable to support the complexity or volume of its business. Understanding and maintaining relationships between customers, suppliers, and products can be a challenging task, especially as an organization grows.

Your task is to revamp and update the software systems of the WRLRR to improve various business processes. Closely studying business practices allows you to identify areas for improvement as well as identify challenges faced by the business. You must look for potential improvements to the overall operation of the business. Examine transactions and returns, report generation, equipment order placement, and other features to support the business. With organization and operation in mind, new systems must be developed to be both scalable and flexible. This will ensure they can be easily updated and maintained to ensure proper function and longevity. Developing a robust software system will enable the WRLRR team to manage operations with greater efficiency, ease, and precision. Such a system can facilitate WRLRR's continued success and growth by working more effectively for both customers and employees.

2 Part 2: Project Deliverables

2.1 Project Deliverables for a Database Course

Educational projects are often made up of deliverables. A deliverable is instrumental in learning how to apply concepts and methodological approaches related to a particular task. That is, an educational objective is reinforced by the corresponding deliverable (Piccinini, & Scollo, 2006). Each of the following deliverables is designed to provide hands-on experience with critical skills learned in the class.

Suggested project deliverables for a Database Design class include those in Table 1.

Table 1. Suggested Project Deliverables for a Database Course

Deliverable Number	Deliverable Goal
1	Entity-Relationship Model
2	Normalized Relational Schema
3	Database Tables
4	Query Design
5	Advanced Query Design
6	Stored Procedures
7	Triggers

2.1.1 Deliverable 1 – Entity-Relationship Model

An Entity-Relationship model provides a precise description of the nature and uses of data within an organization. Its purpose is to reduce real-world complexities to more easily understood abstractions that define entities and the relations among them. An E-R model consolidates the different views of data as perceived by designers, programmers, and users at the conceptual level.

The E-R Diagram should be developed through a step-by-step process. The steps that will be followed are listed below:

- PART A:
    - A listing of objects (potential entities)
        - One good approach is to develop this list on a process-by-process and/or report-by-report basis. Then, consolidate the list, eliminating redundancies.
        - Forms, reports, receipts, etc., are generated by the system and typically are not objects. However, forms, reports, receipts, etc. can provide the designer with information about specific attributes of the objects with which they are associated.
    - A listing of attributes for the entities and composite entities.
- PART B:
    - Develop a Chen or Crow's Foot E-R diagram focusing on entities, relationships, and connectivity. Do not show attributes. Later, cardinality, optional/mandatory, weak entities, etc., may be required, but not for this iteration.

This deliverable is related to educational objectives such as

- The student will be able to analyze database requirements to determine the entities involved in the system.
- The student will be able to analyze database requirements to determine the relationships between entities.
- The student will be able to develop the logical design of the database using data modeling concepts like entity-relationship diagrams.
- The student will be able to implement business rules and understand their impact on the design of a database.

2.1.2 Deliverable 2 – Normalized Relational Model

An E-R model is produced during the conceptual data modeling phase of the database development process. These models can typically be transformed and enhanced through normalization principles. Performing normalization during E-R model development can improve the conceptual model and speed its implementation.

Normalization provides a formal framework for analyzing relation schemas. This analysis is based on their keys and the functional dependencies among their attributes. The normalization process converts complex data structures into simple stable data structures. This is done to reduce data redundancy in tables and improve data integrity. The relational model should be developed through a step-by-step process. The steps that will be followed are listed below:

- Review the identification of attributes and the E-R diagram from Deliverable I. If modifications are needed (as indicated in discussions with your instructor) please include those.
- Using the E-R model and the entities, relationships, and their attributes from Deliverable I, determine and specify the relations. Provide the relational schema for each relation.
- Normalize each relation. Explain the introduction of any new attributes, and why each relation does or does not need to be normalized.
- Indicate the primary key in each relation by underlining. Double-underline or italicize foreign keys.
- Turn in a relational schema for each relation in this format:

    - TABLE_NAME (<u>KEY_ATTRIBUTE$_1$</u>, ATTRIBUTE$_2$, ...ATTRIBUTE$_x$)

This deliverable is related to educational objectives such as

- The student will be able to apply the normalization approach to remove database anomalies to improve database functionality.

2.1.3 Deliverable 3 – Table Creation/Data Entry

The next step in the database lifecycle is implementation and loading. Implementation involves the construction of a database and its tables according to the specification of the normalized relational schema. Data is then loaded into the database tables. Tables must be developed and populated with data before the database can be used.

At this point, the instructor may provide you with a relational model. This helps ensure that everyone has an equal chance for success. It also standardizes future deliverables to make consultations with the instructor more efficient. You are required to use the solution provided. If you spot errors or oversights, consult the instructor.

Using the provided solution, create a database and the appropriate tables. Use the same table names and attribute names as those in the provided relational model.

Your instructor may provide you with a .sql file containing SQL scripts and sample data to use to populate your newly created tables. Be sure that your database includes tables representing both the entities and the composite or associative entities. This data will also populate the composite tables to provide the linkage between the tables being bridged. Your table and attribute names must match those in the provided relational model for the sample data to be useful.

If the instructor does not provide a data set for your use, please request it. This data will make it easier to verify that your results are correct.

The steps that will be followed are listed below:

- Using your relational model, create each table.
- Your numeric keys should use the auto-increment feature.
- Specify table linkages using the designer tool or manually.
- While your tables must include foreign keys, it is not necessary to specify foreign key constraints in this project. While specifying foreign key constraints is a good practice in real life, it can make testing your database difficult. Foreign key constraints are specified using clauses such as "ON DELETE CASCADE" or "ON UPDATE CASCADE ON DELETE RESTRICT".
- Generate a pdf file (or files) containing the following:
    - Print each table and include it in your deliverable.
        - phpMyAdmin steps:
            - click table name in left-hand menu
            - capture image using tool of your choice
        - MySQL Workbench steps:
            - Object Browser, right click table name, select Edit Table Data
        - An image capture tool like Snipping Tool can be used to capture an image of each table.
    - Print the table structure.
        - phpMyAdmin Steps:
            - click table name in left-hand menu
            - click the Structure tab from top menu
            - capture image using tool of your choice

        - MySQL Workbench Steps:
            - Object Browser, right click table name, select Alter Table...
            - In the resulting window, click Columns
        - Again, you may need the snipping tool to capture the table structure.
    - Print the table relationships.
        - phpMyAdmin Steps:
            - click the database name in the left column
            - click the Designer tab in the top menu
            - rearrange the table schemes
            - click Create Relationship in the left menu bar
            - click Save every time you make any single change
            - you may have to leave Designer and re-enter it to see your relationship lines appear
        - MySQL Workbench Steps:
            - Database, Reverse Engineer, Next, Next, <select schemata> Next, Next, Execute, Next, Finish

This deliverable is related to educational objectives such as

- The student will be able to install an RDBMS and utilize a wide range of its features.
- The student will be able to create database and table structures using the SQL data definition commands.
- The student will be able to use SQL data management commands to insert, delete, and update data within database tables.

2.1.4 Deliverable 4 – Query Design

Using the tables developed earlier, develop a set of queries as specified in the query list in Table 2.

Table 2. Query List for Deliverable 4

Query Number	Query Goal
1	List all products in the inventory category table whose manufacturer name starts with an "A" and has a third letter "r". • Include the concatenated category manufacturer and category model name followed by the category type. • The category manufacturer and category model name should take the form "[Category Manufacturer] [Category Model Name]" through concatenation. • Give the concatenated category manufacturer and category model name the alias 'Item'. • Your answer should include Aeros, Aircross, and Airush.
2	List all salesperson information in descending order by sales total. • Include salesperson number, last name, first name, sales count, and sales total (salespersonSalesAmount).
3	List information for each sales transaction. • Include customer number, customer name, sales agreement number, and sale price. • The customer name should take the form "[First] [Last]" through concatenation and should include an alias. • Give the concatenated name a meaningful alias.

4	List information for all sales transactions having a sales transaction date of June 21, 2023. • Include the customer name, serial number, category number, category description, and sales transaction date. • The customer name should take the form "[First] [Last]" through concatenation and should include an alias. • The category description should take the form "[Category Type]: [Category Manufacturer] [Category Model Name]" through concatenation.
5	List all orders that were placed over 1 month ago and have not been received. • Include order number, order date placed, order date received, supplier number, supplier name, supplier contact, and supplier phone. • List by order date placed from oldest to most recent. • CURRENT_DATE and INTERVAL may be useful here.
6	List all customers who have tried to return an item beyond 90 days from purchase and had a refund declined. • Include customer number, customer name, return reason, return condition, and category description. • The customer name should take the form "[First] [Last]" through concatenation and should include an alias. • The category description should take the form "[Category Type]: [Category Manufacturer] [Category Model Name]" through concatenation. • Sort the list in ascending order by customer number.

Any queries that make use of a Join operation should use the WHERE clause or the INNER JOIN. Points will be deducted for queries that are autogenerated by the RDBMS, or that use nested INNER JOINs.

Include data in your tables to test each query (your instructor may provide data for you to use). Be sure to save the queries and submit the database or database location. Provide printouts of both the SQL view and the query result. Be sure that there is data in your tables so that the instructor can duplicate your results!

Submit the queries in a pdf document.

For each query, provide a printout of the SQL as well as the query result.

Save your queries as views.

This deliverable is related to educational objectives such as

- The student will be able to use SQL data query commands to explore database contents and convert data to information.
- The student will be able to apply JOIN operations to combine related rows from two tables into single virtual tables.
- The student will be able to explain why the JOIN is the power behind a relational database.
- The student will be able to sort the results of a SELECT statement.
- The student will be able to remove duplicates from the results of a SELECT statement.
- The student will be able to create and use VIEWs.
- The student will be able to store the results of a query in a VIEW.

2.1.5 Deliverable 5 – Advanced Query Design

Using the tables developed earlier, develop a set of queries as specified in the query list in Table 3.

Table 3. Query List for Deliverable 5

Query Number	Query Goal
7	Provide information about any product categories that were supplied by multiple suppliers. • List the category number, category description, and number of suppliers. • The category description should take the form "[Category Type]: [Category Manufacturer] [Category Model Name]" through concatenation. • Use the GROUP BY clause with a HAVING clause. • This solution requires a DISTINCT. • If an item is supplied multiple times by the same supplier, it should count the number of suppliers, not orders.
8	List all sales in the past two years. • Include sales date, customer number, customer name, category number, category description, and item sales price. • The customer name should take the form "[First] [Last]" through concatenation and should include an alias. • The category description should take the form "[Category Type]: [Category Manufacturer] [Category Model Name]" through concatenation. • CURRENT_DATE and INTERVAL may be useful here. • Sort the output by sales date.
9	List the top 3 best-selling items. • Include category number, category, and total sales. • Use ORDER BY and LIMIT (or TOP). • Use a nested query.
10	List all suppliers for which there are no current orders. • Include supplier number and supplier name. • Sort the list in ascending order by supplier name. • Use a nested query.
11	Provide information about the inventory category that has the highest number of sales. • Include the category number, category description, and the total number of sales. • The category description should take the form "[Category Type]: [Category Manufacturer] [Category Model Name]" through concatenation. • Do not simply order the list and select the top item. Use a nested query.
12	List all attempts to return an item beyond the 90-day return window. • Include the return number, item number, customer number, return date, and purchase date. • Sort the list in ascending order by return number. • Use DATEDIFF to calculate the span between the return date and the purchase date. • While Query 6 performed a similar function, this version is more complex. • A nested query may be needed.
13	List all credit cards associated with a customer. • Use a prepared statement. • Accept a customer number via a variable. • Include customer number, customer name, card number, card type (Visa, Mastercard, etc.), and expiration date. • The customer name should take the form "[First] [Last]" through concatenation and should include an alias. • Deallocate at the end of the query.

14	Provide information about the discount retailer to which the returned item was sent. • Use a prepared statement. • Accept a customer number via a variable. • Include the discount retailer name, discount retailer contact, and discount retailer email. • Deallocate at the end of the query. • Create the prepared statement as a stored procedure that includes a parameter (itemNumParam) to which the placeholder can be compared.

- Submit the queries in a pdf document.
- For each query, provide a printout of the SQL as well as the query result.
- Save your queries as views, except for the last two that involve prepared statements.
    - Prepared statements cannot be saved as views. SQL views are based on SELECT statements, and a prepared statement is based on a PREPARE statement.
    - Save the final two queries as stored procedures.

This deliverable is related to educational objectives such as

- The student will be able to apply SQL aggregate functions.
- The student will be able to group rows when aggregating information.
- The student will be able to filter out rows that do not belong to specified groups.
- The student will be able to embed SQL queries in other queries to narrow the scope of the main query.
- The student will be able to apply advanced SQL statements to change the structure of a database table for normalization.
- The student will be able to apply advanced SQL statements to copy selected table columns from a table.
- The student will be able to apply advanced SQL statements to designate primary and foreign keys.
- The student will be able to JOIN related rows and non-matching rows from two tables into single virtual tables.
- The student will be able to JOIN a table to itself.

2.1.6 Deliverable 6 – Stored Programs

Using the tables developed earlier, develop the stored programs as specified in Table 4.

- Submit the stored programs in a pdf document.
- For each stored program, provide a printout of the code as well as the stored program result.

Table 4. Stored Programs List for Deliverable 6

Stored Program Number	**Query Goal**
1	Develop Stored Procedure • Write a stored procedure to set the value of the reorderNecessary attribute of the inventoryCategory table. ○ It should be set to true when the categoryQuantityOnHand is less than or equal to the categoryReorderQuantity. This condition indicates the item should be reordered. ○ It should be set to false if the categoryQuantityOnHand is greater than the categoryReorderQuantity, meaning the product should not be reordered. • In preparation for developing this sProc, modify the inventoryCategory table to include a new

Boolean attribute called reorderNecessary.

- While it is an unnecessary derived attribute, it is needed for this exercise.
- This attribute should be set to true when an item needs to be reordered.
- The attribute should default to false.
- The default value may not be valid for all the data in your table.
- Therefore, write a stored procedure to set the value of the new reorderNecessary attribute to its correct value.

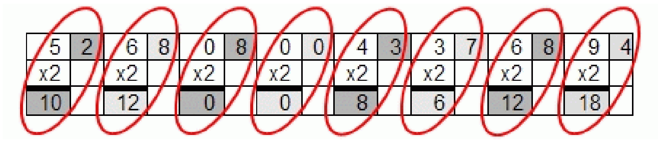
- If the categoryQuantityOnHand is less than or equal to the categoryReorderQuantity, set reorderNecessary to true.
- Otherwise set reorderNecessary to false.
- While multiple approaches can be used, it seems most straightforward to use a pair of UPDATE statements.
- You can test your stored procedure by altering one or more of the inventory counts, and then running the sProc.
- You can alter the counts manually, or by using a statement like

- These steps can be used to test a stored procedure in MariaDB:
  - Click on the database name in the top line.
  - Click the "Routines" tab.
  - Click "Execute" next to the stored procedure name.
- Another way to test a stored procedure follows:
  - In the SQL window enter a command like

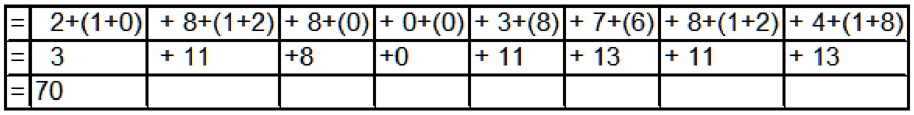
2

Convert Existing Stored Procedure to Stored Function

- In the next stored program, you are provided with an existing stored procedure to convert to a stored function.
- When an item is purchased by credit card, the card number must be validated. The Luhn algorithm to accomplish this is fairly straightforward and consists of three steps. These steps are performed by working from the rightmost digit of the credit card number.
  - Step 1:Begin with the second digit from the right. Double the value of alternate digits of the primary account number. The right-most digit is the check digit.



  - Step 2: Continue working from the right. Add the individual digits comprising the products from Step 1 to each of the unaffected digits in the original number. For example, if the product is 12 then add 1 + 2 to the unaffected digit to the right.



=	2+(1+0)	+ 8+(1+2)	+ 8+(0)	+ 0+(0)	+ 3+(8)	+ 7+(6)	+ 8+(1+2)	+ 4+(1+8)
=	3	+ 11	+8	+0	+ 11	+ 13	+ 11	+ 13
=	70							

  - Step 3: The total from Step 2 must be a number ending in zero (or mod 10 = 0) for the account number to be validated.

70 mod 10 = 0 ∴ Card number is valid

- Convert the following SQL Server sProc to validate a credit card into a MySQL stored function.

- Here is an ISNUMERIC function that you can use:

```
CREATE PROCEDURE prValidateCreditCard (@ccNum char(16), @result BIT) AS
BEGIN
  DECLARE @counter INT, @sum INT, @number INT, @tmp INT
  SET @result=0
  IF @ccNum IS NOT NULL AND len(@ccdNum)=16
    SET @counter=1
    WHILE @counter<=len(@ccNum) AND ISNUMERIC(substring(@ccNum,@counter,1))=1
      BEGIN
        SET @counter=@counter+1
      END
  IF @counter > len(@ccNum)
    BEGIN
      SET @result=0
    END
  ELSE
    BEGIN
      SET @sum=0
      SET @number=0
      SET @counter=len(@ccNum)
      WHILE @counter>0
        BEGIN
          IF @counter>1
            BEGIN
              SET @tmp=(ASCII(substring(@ccNum,@counter-1,1))-48)*2
              IF @tmp>9
                SET @sum=@sum+@tmp-9
              ELSE
                SET @sum=@sum+@tmp
            END
            SET @number=@number+(ASCII(substring(@ccNum,@counter,1))-48)
            SET @counter=@counter-2
          END
          SET @sum=(@sum+@number) % 10
          IF @sum=0
            BEGIN
              SET @result=1
            END
    END
END
GO
```

- Test your stored function with a valid credit card number like 5268080043376894.
    - You can test a stored function like this:

- OPTIONAL: Credit card numbers in the database do not include dashes. You can increase the robustness of the stored function by allowing it to accept credit cards with dashes. You can simply strip the dashes if necessary. If your parameter is named ccNum, you can sanitize the input by including a statement like

	SET ccNum = REGEXP_REPLACE(ccNum,'-',''); ○ Test it with a valid credit card number like 5268-0800-4337-6894.

This deliverable is related to educational objectives such as

- The student will be able to define transactions in SQL and will be able to explain the importance of transactions.
- The student will be able to list and explain the fundamental concepts of stored programs in SQL.
- The student will be able to develop stored programs in SQL
- The student will be able to determine when it is appropriate to use stored programs.

2.1.7 Deliverable 7 – Triggers

Using the tables developed earlier, develop the triggers as specified in the trigger list in Table 5.

- Submit the triggers in a pdf document.
- For each trigger, provide a printout of the code as well as the trigger result.

Table 5. Trigger List for Deliverable 7

Trigger Number	Query Goal
1	Inventory Management • The first trigger represents a typical inventory management task. In typical point-of-sale systems, whenever an item is restocked the associated inventory count is incremented. Whenever an item is sold the associated inventory count is decremented. Upon each change, the trigger compares the inventory count to the reorder quantity by doing the following: ○ The trigger should be fired whenever the categoryQuantityOnHand or categoryReorderQuantity attributes of the inventoryCategory table are updated. ○ A BEFORE UPDATE trigger seems to work better. ○ If the inventory count has reached or fallen below the reorder quantity, then the reorderNecessary attribute is set to true. ○ If the inventory count exceeds the reorder quantity, then the reorderNecessary attribute should be set to false. ○ Note that incrementing or decrementing the inventory count occurs outside the scope of this trigger. ○ You can test this trigger with a query like the following: UPDATE inventoryCategory SET ` categoryQuantityOnHand `= 0 WHERE `categoryNum` = 4 ○ Reminder #1: DO NOT use UPDATE statements in an UPDATE trigger to update the table associated with that trigger. ▪ A table's trigger that is fired on an UPDATE event cannot contain an UPDATE command that modifies that table. ▪ An update trigger will fire each time any row (or rows!) is updated. • A trigger containing an UPDATE to its table is fired on a BEFORE UPDATE event, meaning the trigger will fire infinitely. These are known as recursive triggers. ○ Reminder #2: Within the body of a trigger, the NEW and OLD keywords refer to the table associated with the trigger. ▪ NEW.column_name refers to one of the following: • a column of a new row before it is inserted • an existing row after it is updated. ▪ You typically do not refer to the table name itself in the trigger.
2	Credit Card Validation • This trigger calls the credit card validation stored function developed for the preceding deliverable. When an item is purchased by credit card, the card number must be validated. ○ Be sure to check that the payment type is credit card. ○ Write a pair of triggers to execute the stored function to validate card numbers that you developed. ▪ This trigger should run when salesAgreementCardNumberUsed is inserted or

salesAgreementCardNumberUsed is updated.
        - Reminder: The credit card number is evaluated only when the transaction payment type is credit card.
    - If the credit card is invalid, you should SIGNAL an error.
    - You can test this trigger with queries like the following (you may need to tweak the transaction numbers):

```
--PASS--
START TRANSACTION;
INSERT INTO transaction (transactionNum, transactionType, transactionDate,
transactionTime, customerNum, employeeNum)
VALUES (10000025, 'sale', '2024-04-16', '00:00:00', 1, 3);
INSERT INTO saleTransaction (salesAgreementNum, salesAgreementPaymentType,
salesAgreementCardNumberUsed)
VALUES (10000025, 'Credit Card', '5268080043376894');
COMMIT;

--FAIL--
UPDATE saleTransaction
SET salesAgreementCardNumberUsed = '5268080042376894'
WHERE salesAgreementNum = 10000025;
--PASS--
UPDATE saleTransaction
SET salesAgreementCardNumberUsed = '5268080043376894'
WHERE salesAgreementNum = 10000025;

--PASS--
START TRANSACTION;
INSERT INTO transaction (transactionNum, transactionType, transactionDate,
transactionTime, customerNum, employeeNum)
VALUES (10000026, 'sale', '2024-04-16', '00:00:00', 1, 3);
INSERT INTO saleTransaction (salesAgreementNum, salesAgreementPaymentType,
salesAgreementCardNumberUsed)
VALUES (10000026, 'Cash', NULL);
COMMIT;

--FAIL--
START TRANSACTION;
INSERT INTO transaction (transactionNum, transactionType, transactionDate,
transactionTime, customerNum, employeeNum)VALUES (10000027, 'sale', '2024-04-16',
'00:00:00', 1, 3);
INSERT INTO saleTransaction (salesAgreementNum, salesAgreementPaymentType,
salesAgreementCardNumberUsed)
VALUES (10000027, 'Credit Card', '5268080042376894');
COMMIT;
```

This deliverable is related to educational objectives such as

- The student will be able to define transactions in SQL and will be able to explain the importance of transactions.
- The student will be able to list and explain the fundamental concepts of triggers in SQL.
- The student will be able to develop triggers in SQL
- The student will be able to determine when it is appropriate to use triggers.

2.2 Project Deliverables for a Systems Analysis and Design Course

In many Systems Analysis and Design projects, students are required to first model the business's current system. They then model the requested system that incorporates any feature additions, deletions, or modifications desired by the company. As explained by Spurrier and Topi (2021), "Given that one of the major focal points of systems analysis and design is organizational transformation, it is important to remember that SA&D processes need to deal with both the current reality of the organization (current state) and the desired end state after the organization and/or its way to conduct its business have been transformed (future state)" (p. 43).

Suggested project deliverables for a Systems Analysis and Design course that covers the Unified Process and the Unified Modeling Language include those in Table 6.

Table 6. Suggested Project Deliverables for an SA&D Course

Deliverable Number	Deliverable Goal
1	Use Case Diagram (specific subsystems)
2	Use Case Narrative (specific subsystems)
3	Class Diagram
4	Decision Tree
5	Activity Diagrams (specific subsystems)
6	Sequence Diagram (specific subsystems)

2.2.1 Deliverable 1 – Use Case Diagram (current system)

There are several tools available for gathering and analyzing user requirements, including UML use case diagrams. Use case diagrams are tools for modeling the behavior of a system and helping to capture and specify system requirements. A collection of use case diagrams models the functional requirements of a system. A complete set of use cases specifies all the ways the system is used. It also defines all behavior required of the system.

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. Use cases and actors describe what the system does and how actors use it. Use cases do not describe the internal operation of the system.

Based on examples in the class notes, develop use case diagrams for the following customer-facing business processes:

- Overall system
- Customer purchases item(s)
- Customer returns item(s)
- Generate purchase agreement

You may wish to follow these steps:

1. Define the actors
2. Define their goals
3. Define the system's use cases
4. Identify scenarios
5. identify the relationship between
   a. actors and actors
   b. use cases and use cases
   c. actors and use cases
6. Create use case diagrams for every goal.

This deliverable is related to educational objectives such as

- The student will be able to develop use case diagrams to capture the functional requirements of the system.
- The student will be able to develop use case diagrams to represent the goals of systems and users.
- The student will be able to develop use case diagrams to specify the context in which a system should be viewed.

2.2.2 Use Case Narrative

Use case diagrams are often augmented by use case narratives. Use case narratives are textual descriptions of interaction logic between actors within a domain, often between user and system. Also called use case descriptions or use case specifications, use case narratives can contain acceptance criteria. Use case narratives contain all the information needed to document the functionality of the business processes. They also include all the information needed to build the structural and behavioral diagrams that follow. In addition, they express the information in a less formal way that is usually simpler for users to understand.

Based on examples in the class notes, develop a use case narrative for the following business processes:

- Customer purchases item(s)
- Customer returns item(s)

Each use case narrative should include the following:

- Use Case Name
- Brief Description
- Level
- Trigger(s)
- Primary Actor
- Additional/Supporting Actors
- Stakeholders
- Preconditions
- Main Success Scenario (or Basic Flow)
- Extensions (or Alternate Flows)
- Post-Conditions
    - Success End Condition
    - Minimal Guarantees
    - Failure End Condition
- Frequency
- Special Requirements

You may wish to follow these steps:

1. Pick a high-priority use case and create an overview:
    a. List the primary actor
    b. Determine its type (overview or detail; essential or real)
    c. List all stakeholders and their interests
    d. Determine the level of importance of the use case
    e. Briefly describe the use case
    f. List what triggers the use case
    g. List its relationship to other use cases
2. Fill in the steps of the normal flow of events required to complete the use case
3. Ensure that the steps listed are not too complicated or long and are consistent in size with other steps
4. Identify and write the alternate or exceptional flows
5. Carefully review the use case description and confirm that it is correct
6. Iterate over the entire set of steps again

This deliverable is related to educational objectives such as

- The student will be able to develop use case narratives to more fully document the functionality of individual use case diagrams.

2.2.3 Deliverable 3 – Class Diagram

Many approaches employ use case diagrams to describe the high-level requirements, and class diagrams to describe the domain. Class diagrams can help with requirements discovery. They bring clarity and explicitness of understanding that helps provide analysts with insight into software requirements.

The class diagram can be used for modeling the internal software design of object-oriented software applications. When used in domain modeling, class diagrams should focus on the real-world domain of interest, its concepts, and their relationships.

For this deliverable, you will develop a complete class diagram to model the WRLRR system. The class diagram should include the following:

- Classes/Entities
- Attributes
- Associations/Relationships with Multiplicities/Cardinalities shown for each association or relationship

Save the class model as a PDF and turn it in. Be sure to include any additional clarifying narrative descriptions such as any assumptions you had to make.

You may wish to follow the approach below:

- Identify classes
- Identify the attributes
- Identify the relationships (association, aggregation, composition, and generalization), including multiplicity

This deliverable is related to educational objectives such as

- The student will be able to develop class diagrams representing the system's classes, their attributes, operations, and the relationships between objects.

2.2.4 Deliverable 4 – Decision Tree

Process specifications can be used to provide a more detailed specification of the process logic of system procedures for programmers. One method used in developing process specifications is the decision tree.

Decision trees are particularly useful for expressing the logic required by complex conditions. A decision tree is a graphic documentation tool that represents conditions and their resulting action. Decision trees are used when complex branching occurs in a structured decision process. Decision trees are also useful when it is essential to keep a string of decisions in a particular sequence.

Develop a decision tree to depict the logic involved in the decision to issue a refund.

This deliverable is related to educational objectives such as

- The student will be able to express the logic required by complex conditions using various approaches.

2.2.5 Deliverable 5 – Activity Diagrams (specific subsystems)

Activity diagrams can be used in all stages of software development and for various purposes. High-level activity diagrams can capture business processes and activities at a higher level of abstraction than other diagrams.

Activity diagrams can model business requirements, create high-level views of system functionalities, analyze use cases, and for various other purposes.

An activity diagram is used by developers to understand the flow of programs on a high level. An activity diagram can illustrate high-level business processes to primitive algorithms.

Based on examples from class, develop an activity diagram (with swim lanes) for the following business processes:

- Customer purchases item(s)
- Customer returns item(s)

You should include swim lanes for (at least) Customer, Employee, System, and Payment Processor.

This deliverable is related to educational objectives such as

- The student will be able to develop UML activity diagrams. These provide a view of the behavior of a system by describing the sequence of actions in a process.

2.2.6 Deliverable 6 – Sequence Diagrams (specific subsystems)

A sequence diagram is a dynamic model depicting the objects that participate in a single use case. It includes the explicit sequence of messages that are passed between those objects over time. Sequence diagrams are useful for understanding real-time specifications and complex use cases. This is because they emphasize the time-based ordering of the activity that takes place among a set of objects.

An analyst develops a set of instance sequence diagrams, each of which depicts a single scenario within the use case. Sequence diagrams can help capture required objects and classes involved in a scenario. They can also model the order of interactions between objects involved in carrying out the functionality of the scenario.

Based on examples in the class notes, develop a sequence diagram for the following business processes:

- Customer purchases item(s)
- Customer returns item(s)

This deliverable is related to educational objectives such as

- The student will be able to develop UML sequence diagrams. These show the objects that participate in a use case. They also show the explicit sequence of messages passed between those objects over time.

2.3 Additional Course Applications

This case has been designed to provide a modular project-based case for software development. Example projects were provided for a database course and an analysis and design course. However, the case could potentially be used in a variety of courses. The possible deliverable lists included in the tables below are intended to provide only a starting point for other courses. Their inclusion is solely to provide instructors with ideas of how the case study could be used in additional courses.

2.3.1 Capstone Systems Development Course

The case is applicable in a capstone CIS/IS systems development or software engineering course. The associated project might require deliverables such as those in Table 7, as proposed by Russell and Russell (2015).

Table 7. Capstone Systems Development Project Deliverables

Project Phase	Deliverable
Planning	Project Plan, Gantt Chart and PERT Chart
	Function Point Analysis Report
	Cost-Benefit and Break-even Analysis Report
Analysis	Functional Business Model (Use Case Diagram, Use Case Descriptions, and Activity Diagrams)
	Structural Model (Class Diagram)
	Systems Proposal Report and Presentation
Design	Interface and Navigation Design
	Behavioral Model (Sequence Diagrams and State Machine/Transition Diagrams)

	Database Design and Program Design
	Structured Walkthrough
Implementation	Program Development
	Program Testing
Final	Systems Specification Report
	Systems Specification Presentation by Teams

2.3.2 Front-End Development / UX Design / Web Development Course

The case has also been used in front-end development/UX design/web development. Possible end products are prototypes of various data entry forms developed using HTML, CSS, JavaScript, MySQL, and PHP. Table 8 includes some suggested deliverables to provide a starting point for instructors.

Table 8. Front-End Development Project Deliverables

Project Phase	**Deliverable**
Analysis	User research (includes hypothetical personas, aka user personas, aka user profiles)
	Experience maps (aka journey maps, aka user journeys)
Design	Navigation diagrams
	Wireframes
	Prototypes (order new inventory, process new inventory, employee entry form, customer entry form, sale form, return form)
Security and Accessibility	Interface design security
	Internationalization
	Accessibility
Testing	User testing
	Usability testing

2.3.3 Data Analytics / Business Analytics Course

This case might also provide the foundation for a data analytics or business analytics course project. However, the current data set is too limited in size. A colleague proposed using a test data generator tool to generate synthetic data. Synthetic data sets could enable the analysis of sales trends, return trends, activity popularity, and emerging wind sports opportunities. Collier (2023) outlines a data analytics project (Table 9) based on the data analytics lifecycle (Erl et al., 2016).

Table 9. Data Analytics Project Deliverables

Project Phase	**Description**	**Outcomes**
Business Case Evaluation	Create a well-defined business case with a clear explanation of the justification, motivation, and goals of carrying out the analysis. This helps stakeholders know the business resources that will be utilized and identify expected challenges that will need to be overcome.	Identify key performance indicators to determine assessment criteria and guidance for the evaluation of the analytic results Develop SMART (specific, measurable, attainable, relevant, and timely) goals for the analysis project Determine a budget for the project Identify possible challenges that will need to be overcome.
Data Identification	Identify the sources of datasets, both internal and external, for the analysis project.	List datasets to be utilized in the analysis project, their sources/locations, and any identified challenges for accessing them.
Data Acquisition and Filtering	Gather data from all the data sources identified in the previous stage. Apply automatic filtering to the data to remove corrupt data or data without value to the analysis objectives.	Develop datasets in various forms (e.g., collections of files, database views, API integration, XML documents, etc.) Metadata may be added at this stage.
Data Extraction	Transform data from incompatible sources	Prepare extracted and transformed data for

	into a format that can be used by the analysis tool of choice.	loading.
Data Validation and Cleansing	Establish validation rules and remove any known invalid data. When appropriate, use redundant data from other datasets to fill in missing data.	Produce validated, clean datasets.
Data Aggregation and Representation	Integrate multiple datasets to arrive at a unified view by resolving differences among datasets, e.g., different data structures or semantics.	Produce a single unified, aggregated dataset.
Data Analysis	Perform the analysis using one or more types of analytics or data visualization techniques. Examples include computing an aggregation for comparison, evaluating a pattern over time, intensive data mining, and complex statistical analysis.	Answer research questions or curiosities posed in the business case evaluation stage. This may confirm/disconfirm an expected cause of a phenomenon or report the results of an exploration into the data.
Data Visualization	Use data visualization tools and techniques to communicate the analysis results for effective interpretation by business users.	Develop visual reports so that users can understand and interpret the results of the data analysis stage.
Utilization of Analysis Results	Determine how and where processed analysis data can be further leveraged.	Provide a plan for how and where the analysis results will be utilized to improve organizational operation and/or strategy.

3 Note to Instructors

A more thorough explanation of this case development approach is available in the companion paper (Parker & Chiang, 2024).

Faculty members considering or adopting this case can obtain the teaching note by contacting the author by email. The author is required to verify that the person requesting the teaching note is a bona fide faculty member by checking the IS Faculty Directory, Google Scholar, or the website of the requester's affiliated university.

The teaching note explains how the insertion points in the baseline case above can guide instructors in altering case content. Instructors can insert additional modules to expand the case, differentiating it and customizing it to their individual needs. Links are included in the teaching notes for sixteen additional modules. A set of macros has been developed and made available to automate the process of incorporating modules into the case.

The teaching note provides a solution to the database project deliverables. Additional resources are provided to reduce the instructor's workload. These resources include a script to create the complete database, including tables, table data, queries, triggers, and stored programs. Another script is provided to create only the tables, and another to populate the tables. In addition, a document containing table names, attributes, data types, and properties is provided. The author shares this document with students after they develop and normalize the relational schema but before table creation. When student projects make use of standardized table and attribute names, it is easier to assist students and grade projects.

The teaching note also provides a solution to the systems analysis and design project outlined in the case. While the solution has not been tested as thoroughly as the database solution, it provides basic guidance for instructors. The teaching note even provides a link to a version of a data flow diagram (DFD) solution. Some instructors still embrace the instructional benefits of data flow diagrams. Others, concerned with cybersecurity and threat assessment, appreciate that DFDs provide insights into every instance where data is potentially vulnerable. The inherent usefulness of DFDs as an analysis tool and threat assessment tool has resulted in a recent resurgence.

This is a very versatile case designed for applications across courses. That was one of the design goals from the inception of the case, and it has successfully fulfilled that goal. It has been tested over 10 years in the following courses:

- Systems Analysis and Design
- Database Design and Implementation

- Front-end Development (including UX Design and Web Development)

It is also suitable for use in the following courses:

- Capstone Systems Development
- Data Analytics (including Business Analytics and Data Science)
- Data Warehouses

.

References

Collier, C. A. (2013). Teaching case: Learning skills of the data analytics lifecycle with Microsoft Power BI and national parks data. *Communications of the Association for Information Systems*, *52*, 238-248.

Erl, T., Khattak, W., & Buhler, P. (2016). *Big data fundamentals: Concepts, drivers & techniques*. Prentice Hall Press.

Parker, K. R. & Chiang, C. T. (2024). Modular design of teaching cases: Reducing workload while maximizing reusability. [Unpublished manuscript].

Piccinini, N., & Scollo, G. (2006). Cooperative project-based learning in a web-based software engineering course. *Educational Technology & Society, 9*(4), 54-62.

Russell, J. & Russell, B. (2015). A systems analysis and design case study for a business modeling learning experience for a capstone CIS/IS systems development class. *Information Systems Education Journal, 13*(6), p. 77-96.

Spurrier, G. & Topi, H. (2021*). Systems analysis & design in an age of options*. Prospect Press.

About the Authors

Kevin R. Parker. Dr. Parker is a Professor Emeritus of Informatics and the Founding Chair of the Department of Informatics and Computer Science at Idaho State University. He has an extensive professional and academic background that spans both Information Systems and Computer Science. He has both a Bachelor's and Master's degree in Computer Science, and a Ph.D. in Information Systems, and has taught courses in Information Systems, Computer Science, Information Technology, and Geographical Information Systems. Parker’s research interests include improving core IT courses, the impact of IS developments on curriculum, and emerging systems development issues. Dr. Parker holds a B.A. in Computer Science from the University of Texas at Austin (1982), an M.S. in Computer Science from Texas Tech University (1991), and a Ph.D. in Information Systems from Texas Tech University (1995).