Computer Science 1
Guild Team Programming

Semester Project

Instructor: William A Loring

Table of Contents

Agile Software Development 5
- What is Agile? 5
- DRY Software Engineering 5
- SOLID Software Engineering 6

Guild Team Based Software Engineering 6
- Share Your Experience 7
- Guild Assignments 7
- Blackboard Groups 7
- Discord 7
- GitHub 7

Guild Software Engineering Process 7
- Get Started 8
- Development 8
- Testing 8
- Conclusion 8
- Sample KanBan Board 9
- Assignment Minimum Requirements 9

Guild Shared Software Engineering GitHub Process 9
- Organize 9
- Commits 9
- Before You Commit 10
- When You Commit 10

Example Project Workflow 10
- First 10
- Workflow 11

Week 8 Milestone: Design 11
- POS (Point of Sale) 12
- GUI Design for Python 12
- Assignment Submission 14

Week 10 Milestone 15
Requirements 15
Multiple Table ERD with Bridge Entity 15
Business Rules 16
Assignment Submission 17
Week 11 Milestone 17
Requirements 17
Shared Coding Process 18
Requirements 18
Assignment Submission 18
Week 12 Milestone 19
Requirements 19
Assignment Submission 19
Week 13 Milestone 19
Requirements 19
Assignment Submission 19
Week 14 Milestone 20
Requirements 20
Assignment Submission 20
Week 15 Milestone: 20
Requirements 20
Presentation Requirements 20
Assignment Submission 21
Week 16 Milestone 21
Program Requirements 21
Assignment Submission 21
Finals Week 22
Requirements 22
Assignment Submission 22
Project Ideas 22
Baseball Statistics and Team Tracker 22
Healthcare Management System 25

Public Library Management Program 27
Stock Portfolio Tracker 29
Point of Sale System 31
More Project Ideas 31

Agile Software Development

Agile development is one of the current processes for software development and other development activities.

What is Agile?

Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methods or Agile processes generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals.

https://www.cprime.com/resources/what-is-agile-what-is-scrum

DRY Software Engineering

Don't **R**epeat **Y**ourself (**DRY**) is a principle of software engineering aimed at reducing repetition of software patterns. It you are repeating any code, there is probably a better solution.

The DRY principle is stated as "Every piece of knowledge must have a single, unambiguous, authoritative representation within a system". This means that there shouldn't be anything in your code that is duplicated somewhere else.

Violations of DRY are typically referred to as WET solutions, which is commonly taken to stand for "write every time", "write everything twice", "we enjoy typing" or "waste everyone's time".

Create classes with a single purpose or theme. Abstract as much as possible from the application to the classes. The application contains as little logic as possible. It creates and uses objects and their methods.

SOLID Software Engineering

In object-oriented computer programming, **SOLID** is a mnemonic acronym for five design principles intended to make software designs more understandable, flexible, and maintainable. The principles are a subset of many principles promoted by American software engineer and instructor Robert C. Martin, first introduced in his 2000 paper Design Principles and Design Patterns.

The **SOLID** concepts are:

- **The Single-responsibility principle:** a class should only have a single responsibility, that is, only changes to one part of the software's specification should be able to affect the specification of the class. A class should only have one job.
- **The Open–closed principle:** Classes should be open for extension and closed for modification. A class should be extendable without modifying the class itself. The code should not have to be changed, but new functionality can be added.
- **The Liskov substitution principle:** Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program. This means that, given that class B is a subclass of class A, we should be able to pass an object of class B to any method that expects an object of class A and the method should not give any weird output in that case. This is the expected behavior, because when we use inheritance, we assume that the child class inherits everything that the superclass has. The child class extends the behavior but never narrows it down.
- **The Interface segregation principle:** Many client-specific interfaces are better than one general-purpose interface. Clients should not be forced to implement a function they do not need.
- **The Dependency inversion principle:** Our classes should depend upon interfaces or abstract classes instead of concrete classes and functions.

Guild Team Based Software Engineering

Software engineering is rarely done alone. It is almost always done as part of a team.

The class will be divided into guilds of 2-3 people each. Guild membership is assigned by the Game Master (the instructor).

Each guild should come up with their own name (let's keep these PG-Rated please). Please let me know what your Guild name and number is. I will re name your GitHub repository.

Share Your Experience

In any class, some students, depending upon their major, programming experience, artistic talent, etc. may exhibit more proficiency than others on certain aspects of the assigned course work. A Guild gives an opportunity for everyone on the team to work together and share their experience.

This project is as much about working as a team as it is about the assignment. The process of coming together, helping each other out is a huge part of team-based software engineering.

Guild Assignments

Guild assignments are team projects where everyone works on the same code project.

Guild members are encouraged to be resources for the other Guild members. They can lend a hand if someone gets stuck on a problem. A 2nd eye on the code can find a missing semi colon or other minor error.

Individual assignments are not team projects. Everyone does their own work.

Blackboard Groups

Each Guild Team has a Group area in Blackboard under Guilds. This is used primarily used grading purposes.

Discord

Please use Discord for Guild communication. Please send an invite to the instructor.

GitHub

Each Guild will have a separate shared GitHub Repository. This repository should be used to store any code or text documentation pertinent to that assignment.

Guild Software Engineering Process

This is a highly suggested process that is like working in real world Agile software engineering.

Get Started

- Live scrum (meeting) to create pseudocode.
    - The pseudocode is a text file.
- Divide up coding into small tasks. Put in the KanBan board.
    - Read **GitHub KanBan Board** in Resources in Blackboard.

Development

- Comment your code descriptively.
    - Why did you do this, is there something left to do, did you hard code something for testing,
- Commit small changes early and often.
- Leave detailed commit descriptions.
    - Make sure the code compiles before pushing to GitHub.

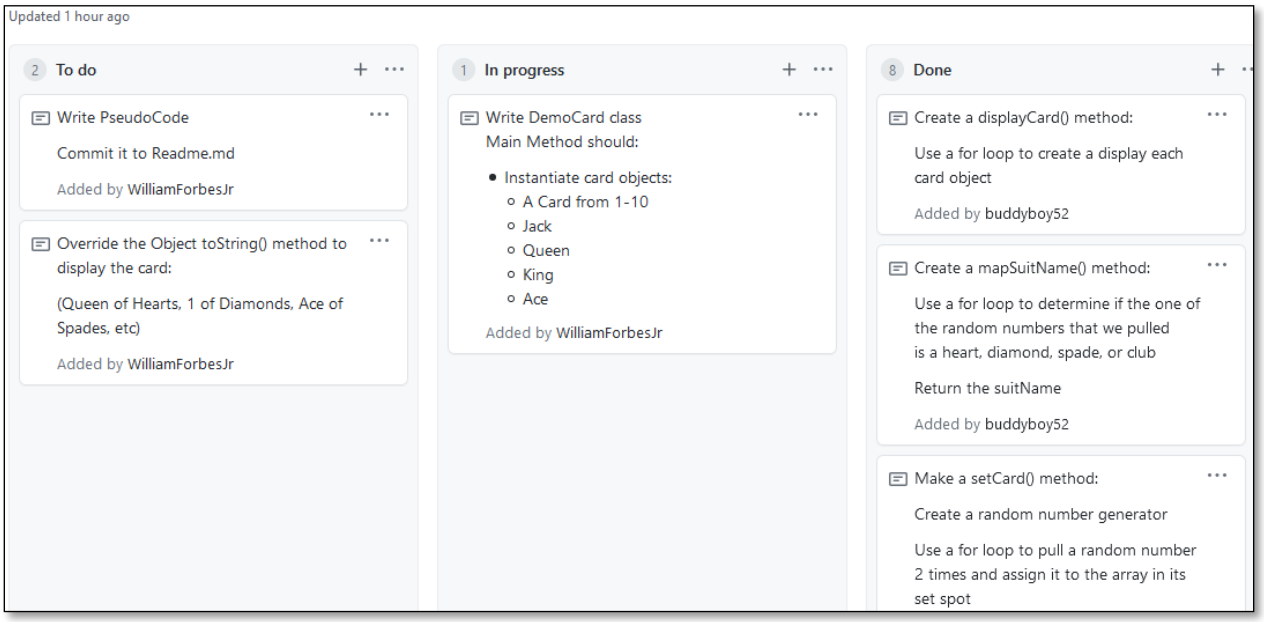
Testing

- Test the code to ensure it meets the requirements.

Conclusion

- Briefly describe who did what.
- Briefly describe your Guild's process.

Sample KanBan Board



Assignment Minimum Requirements

Think of this document as setting out the minimum requirements for the project. As long as the minimum requirements are met in your assignments, you have license to be creative and go further than the assignments request.

Guild Shared Software Engineering GitHub Process

Many of you are brand new to GitHub. We are going to use a basic approach. It requires communication with the team (which isn't a bad thing!).

Organize

- A divide-and-conquer approach is a good way to work on these assignments. When pair programming and working asynchronously, the number 1 thing is to organize the work first so people aren't writing duplicate methods. Even if you are perfectly committing, pushing, and pulling, you'll end up tripping over each other's code.
- Pens shouldn't touch paper until everyone knows exactly who is doing what.

Commits

- Commits should be small and frequent. Add a new method? Commit. Add a new data type? Commit. Have comments to add? Commit.

- Your commit messages should be written to not just say what you did, but why you took an approach. Treat this as your voice to the team, so they almost don't need to talk to you to see why you did something.
- The origin (GitHub) should be the source of truth for your work. If your local work is different, it needs to be committed or overridden with the origin since it's out of date.

Before You Commit

- **Pull**. You want to be sure that you're committing to the most recent version of the branch. Pulling right before committing will help avoid potential merge conflicts.

When You Commit

- **Push** immediately. That ensures that the origin (GitHub) is perfectly in sync with the most recent changes for everyone else. Remember: your commits are for everyone else so they're up to date with the current code.
- Message your group and let them know that you pushed. Yes, we should be pulling before committing, but it's a courtesy and avoids any potential issues.
- If you pull and there are conflicts that need to be resolved. . .
    - Say you pull and your teammate made some changes on that file. The easiest way to resolve this is to first copy your changes somewhere else (these should be small if we're committing correctly)
    - Accept the changes from the pull (Remember GitHub is the source of truth)
    - Add your changes back in
    - Commit then push.

Example Project Workflow

Here is an example of project workflow. You and I are on the same team on this project.

First

- We meet, talk over the pseudo, and setup a plan. You take conversion methods and I take display methods or something.

- Either setup the main class together or whoever can do it right away volunteers. We set it the main class together and push it.

Workflow

- I open GitHub before my IDE to check for updates. No updates.
- I pull and double-check I'm up to date.
- I write a small method displayInches()
- I pull again to check for changes
- Commit my changes. My commit comment says:
    - "Add displayInches method. Right now, it's hard-coded since we don't have a conversion yet but will update once we get data".
- Push
- I send message to the team:
    - Hi everyone I just pushed to add displayInches"
- Repeat

Your combined code is going to have different styles and approaches. That's perfectly okay and expected. When you go and work in the field, you will see a multitude of different styles and approaches in large codebases and will have to get used to that. If someone doesn't agree with a solution or approach, discuss it, and go with what you collectively decide.

Approach the project with the mindset that it is a single application and not a school assignment: 1 project 1 team. These discussions can be one of the bigger challenges of development and something you will regularly do day-to-day.

Week 8 Milestone: Design

100 points

Time Required: 90 minutes

We are going to use Agile development for this project. Please create a new folder for each week’s work. Each week will have a functional deliverable.

This is an open-ended project. There are only a few requirements set out in this project document. At this stage of your learning, you have all the skills and experience necessary to design and create a project of your own design.

Python: OOP GUI, Flask Web - SQL

Java: OOP CLI or GUI - SQL

C++: OOP CLI - SQL

POS (Point of Sale)

This is a suggested direction for your Semester project.

You could also track team sport information, medical clinic, etc.

We are going to create a POS (Point of Sale) system for a small business. You choose the business.

You can build from your program from last semester or any other project that a team member has started.

Theodora wants to track the following items. Start with the simple items, add what you can each week. You do not have to implement all these ideas.

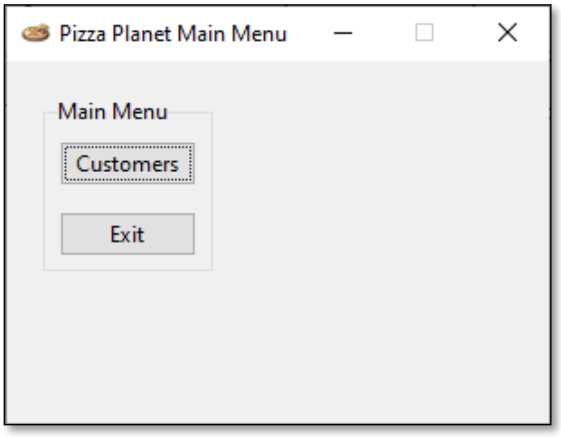
- Persistent data stored in a SQL database.
- Customer information
- Menu of items for sale
- Cash balance: Uncle Fred sold his ukulele for $500. We can use that for purchasing our initial inventory.
- Running inventory of items as they are sold
- Increase inventory when purchased from vendors
- Profit and Loss report
- Random disasters and windfalls
- A back story about the business

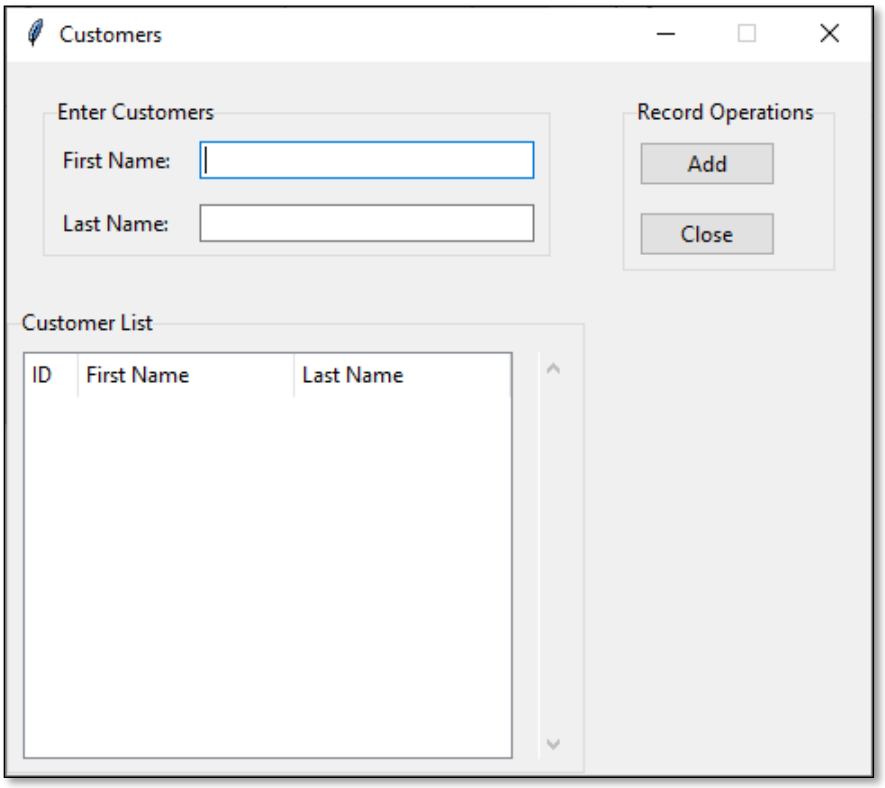
GUI Design for Python

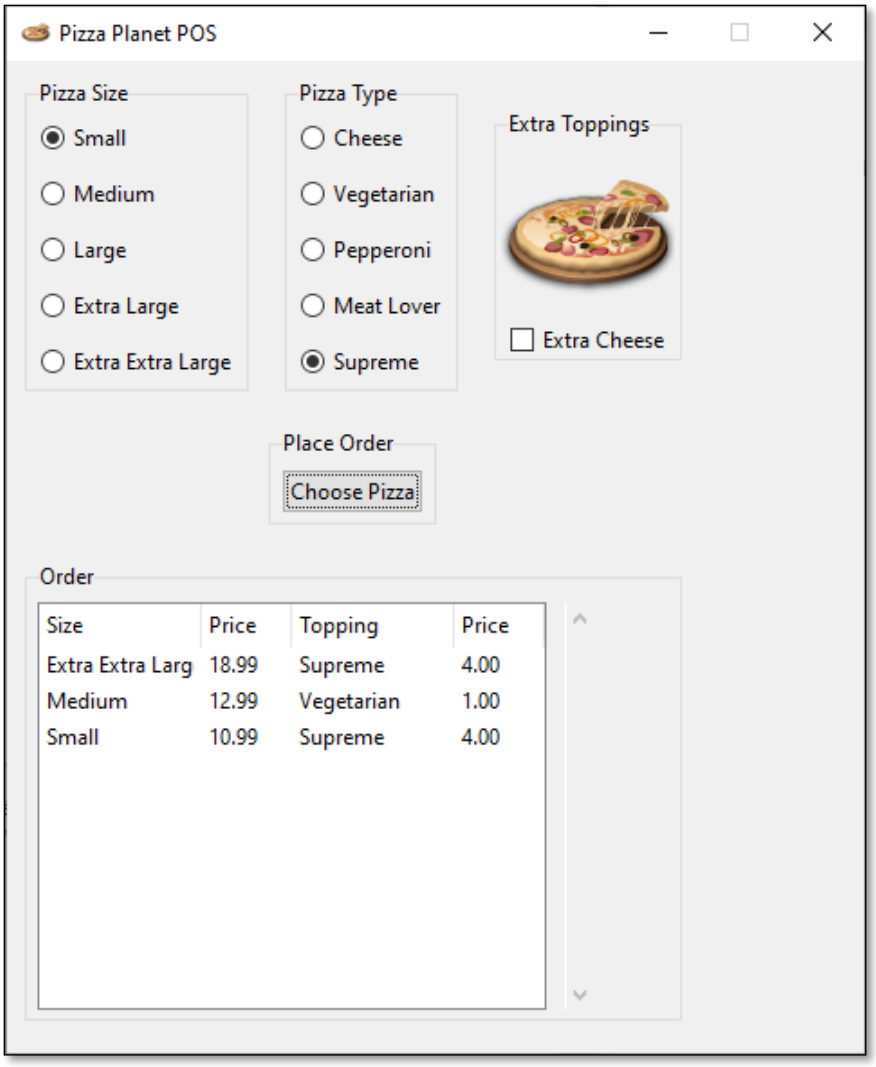
Draw the GUI on paper. We may want several screens, we have too much information for one screen. Each entity may need its own separate screen to modify its information.

- Main screen
- Customer Entry
- Sale Entry
- Inventory Entry

Example run:







Assignment Submission

There is a link for each team.

https://classroom.github.com/a/eNRrJRYk

This is design week. What are you inspired to create?

Project Plan

1. Submit a project plan in Word for your development process and result.
2. Include requirements for your program.
3. Use TODO to sketch out your first iteration of your program.
4. Include a drawing of the interfaces.

1. **Milestone Folder:** Create a Milestone folder for each week's successful iteration of your project. Milestone 8, Milestone 9, etc. That will give you a way to easily go back to a known working version is the next version has trouble.
2. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is created in GitHub.
3. **Guild Team Submission in Blackboard:** Submit a link to a short screencast showing the functionality of the program.
4. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Week 10 Milestone

100 points

Time Required: 90 minutes

Requirements

- Follow and complete the attached tutorial → **Python SQLITE Game Shop with Multiple Tables**

This tutorial will give you much of the SQL code needed to complete your project. You will add interactivity later on.

Multiple Table ERD with Bridge Entity

This is where database planning starts.

We have our entities. Like OOP, entities represent something in the real world we want to keep track of.

- Customers
- Products
- Sales (transaction bridge table)

Create an ERD of your database.

Business Rules

Business rules are how the entities interact. A functional real-life customer sales tracking database would have these business rules.

- A product can be sold to many customers.
- A customer can purchase many products.

This is an example of many to many relationships. You can’t have many to many relationships in SQL. You can have two tables with a bridge or junction table connecting them as shown below to implement the many to many business rules.

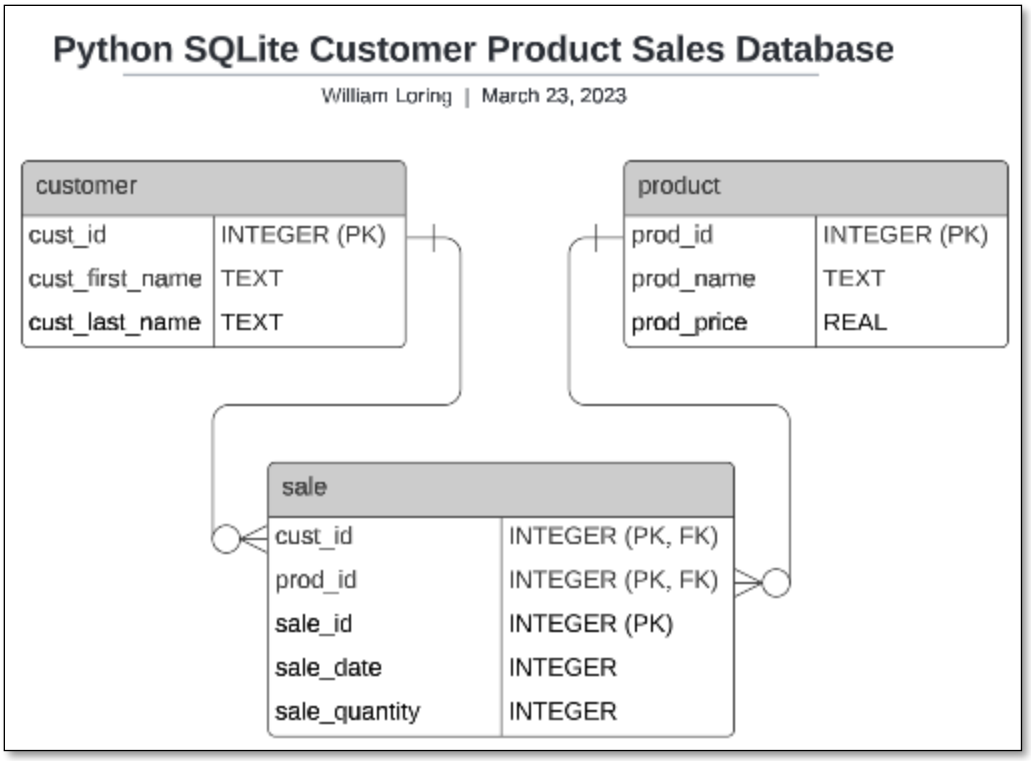
Primary Key: A primary key is a column or a set of columns in a table whose values uniquely identify a row in the table.

Foreign Key: A foreign key is a column or a set of columns in a table whose values correspond to the values of the primary key in another table.

Composite Key: A composite key is made by the combination of two or more columns in a table that can be used to uniquely identify each row in the table when the columns are combined uniqueness of a row is guaranteed, but when it is taken individually it does not guarantee uniqueness, or it can also be understood as a primary key made by the combination of two or more attributes to uniquely identify every row in a table.

These two tables are related through a composite primary key in the sale table. This composite primary key connects the two tables.

This is an example to get you started. You will want to eventually add more fields.



Assignment Submission

We are using Agile development. We want a functioning deliverable each week.

1. **Milestone Folder:** Create a Milestone folder for each week's successful iteration of your project. Milestone 10, Milestone 11, etc. That will give you a way to easily go back to a known working version if the next version has trouble.
2. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is created in GitHub.
3. **Guild Team Submission in Blackboard:** Submit a link to a short screencast showing the functionality of the program.
4. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Week 11 Milestone

Requirements

- Comment each line of code as shown in the tutorials and other code examples.

- Follow all directions carefully and accurately.
- Think of the directions as minimum requirements.

Shared Coding Process

1. Pseudocode first
2. Split up coding tasks
3. KanBan
4. Code and communicate
5. Commit often
6. Test and submit

Requirements

It is up to you to add your planned features to the program.

Please provide evidence of planning, teamwork, and collaboration.

One of the feature requirements is persistent data storage.

SQLite

SQLite allows for more robust data storage. This is one of the most popular methods of storing data. SQL databases are everywhere! You retrieved this assignment from an SQL database. Blackboard uses Java Server Pages and MySQL. WNCC uses Microsoft Active Server Pages and Microsoft SQL.

Assignment Submission

1. **Milestone Folder:** Create a Milestone folder for each week's successful iteration of your project. Milestone 8, Milestone 9, etc. That will give you a way to easily go back to a known working version if the next version has trouble.
2. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is created in GitHub.
3. **Guild Team Submission in Blackboard:** The Guild leader submits a screenshot of their GitHub repository.
4. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Week 12 Milestone

This project uses the same GitHub assignment repository as the last assignment.

Requirements

Next iteration with a working deliverable.

Assignment Submission

1. **Milestone Folder:** Create a Milestone folder for each week's successful iteration of your project. Milestone 8, Milestone 9, etc. That will give you a way to easily go back to a known working version if the next version has trouble.
2. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is created in GitHub.
3. **Guild Team Submission in Blackboard:** Submit a link to a short screencast showing the functionality of the program.
4. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Week 13 Milestone

Requirements

Next iteration with a working deliverable.

Assignment Submission

There is a link for each team.

https://classroom.github.com/a/zIQXnTf8

1. **Milestone Folder:** Create a Milestone folder for each week's successful iteration of your project. Milestone 8, Milestone 9, etc. That will give you a way to easily go back to a known working version if the next version has trouble.
2. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is created in GitHub.
3. **Guild Team Submission in Blackboard:** Submit a link to a short screencast showing the functionality of the program.

4. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Week 14 Milestone

Requirements

Next iteration with a working deliverable.

Assignment Submission

1. **Milestone Folder:** Create a Milestone folder for each week's successful iteration of your project. Milestone 8, Milestone 9, etc. That will give you a way to easily go back to a known working version is the next version has trouble.
2. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is created in GitHub.
3. **Guild Team Submission in Blackboard:** Submit a link to a short screencast showing the functionality of the program.
4. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Week 15 Milestone:

Please read all the directions before beginning the assignment.

Requirements

Next iteration with a working deliverable.

Presentation Requirements

This will be a screencast with each member presenting a part of the program.

1. Describe how you developed your program.
2. Demonstrate the flow of the program.
3. Lessons learned.

Assignment Submission

1. **Milestone Folder:** Create a Milestone folder for each week's successful iteration of your project. Milestone 8, Milestone 9, etc. That will give you a way to easily go back to a known working version if the next version has trouble.
2. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is created in GitHub.
3. **Guild Team Submission in Blackboard:** Submit a link to a short screencast showing the functionality of the program.
4. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Week 16 Milestone

Program Requirements

Next iteration with a working deliverable.

- Polish your program. Make it look nice.
- Test everything.
- Comment the code so anyone can understand your program.
- Reorganize your code to optimize it.

Assignment Submission

1. **Milestone Folder:** Create a Milestone folder for each week's successful iteration of your project. Milestone 8, Milestone 9, etc. That will give you a way to easily go back to a known working version if the next version has trouble.
2. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is created in GitHub.
3. **Guild Team Submission in Blackboard:** Submit a link to a short screencast showing the functionality of the program.
4. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Finals Week

Final submission of your application.

Requirements

A fully functioning, polished, full stack application.

Assignment Submission

1. **Guild GitHub Assignment:** The Guild pseudocode, KanBan board, and code is stored in GitHub.
2. **Guild Team Submission in Blackboard:** Submit a link to a short screencast showing the functionality of the program.
3. **Guild Individual Evaluation:** Each Guild member submits an Individual Guild Evaluation in Blackboard.

Project Ideas

Baseball Statistics and Team Tracker

Project Introduction

In this semester-long project, students will develop a comprehensive Baseball Statistics and Team Tracker application.

Project Goals

1. **Learn Fundamental Concepts**: Gain a solid understanding of programming fundamentals, data structures, and algorithm design.
2. **Apply Object-Oriented Programming**: Develop object-oriented programming skills by creating classes and objects to represent baseball statistics and teams.
3. **Data Management**: Implement data storage and retrieval techniques to manage baseball player statistics and team information.
4. **User Interface Design**: Design an intuitive user interface for data input and retrieval.
5. **Problem Solving**: Enhance problem-solving abilities by addressing specific challenges related to baseball data tracking.

Customer Scenario: Meet John, a Baseball Enthusiast

John is a passionate baseball enthusiast who regularly follows Major League Baseball (MLB) games. He loves to keep track of player statistics, team performance, and player comparisons. However, he finds it tedious to gather and analyze this data manually from various sources. John needs a software solution that allows him to effortlessly track, calculate and analyze baseball statistics.

Project Planning

- Define project scope and requirements.
- Create a project plan with milestones and deadlines.

User Interface Design

- Create a user-friendly interface for data input and display.
- Implement data validation and error handling.

Algorithm Implementation

- Implement algorithms for statistical analysis.
- Enable comparisons between players and teams.

Testing and Debugging

- Conduct thorough testing to identify and fix bugs.
- Ensure data accuracy and software reliability.

Documentation and Presentation

- Document the project's code and functionalities.
- Prepare a presentation on the project's development and features.

Sample Pseudocode

Below is a sample pseudocode snippet for calculating a player's batting average:

```
Function calculateBattingAverage(hits, atBats)
    If atBats > 0
        battingAverage = hits / atBats
    Else
        battingAverage = 0
```

```
    End If
    Return battingAverage
End Function
```

Evaluation and Grading

Students will be evaluated based on:

- Project completion and functionality.
- Code quality and organization.
- Problem-solving skills.
- Presentation of the project.
- Adherence to project milestones and deadlines.

Healthcare Management System

Project Overview

In this semester-long project, students will design and implement a Healthcare Management System (HMS). The HMS will be a software application that facilitates the management of patient records, appointments, and billing for a fictional healthcare facility.

Project Components

Customer Story

- Develop a creative story about a healthcare facility named "HealthyCare Clinic."
- Describe the challenges faced by the clinic in managing patient data and appointments.
- Present the customer's expectations for the HMS.

System Design

- Define the system's architecture and modules.
- Create a high-level flowchart illustrating the software's operation.
- Discuss data storage and security considerations.

User Interface

- Design the user interface of the Healthcare Management System.
- Create wireframes and mockups.
- Consider user experience and accessibility.

Programming

- Write code to implement the HMS functionality.
- Develop features for patient registration, appointment scheduling, and billing.
- Ensure data integrity and error handling.

Testing and Debugging

- Test the HMS thoroughly to identify and fix bugs.
- Conduct user acceptance testing with mock data.
- Ensure the system meets customer requirements.

Sample Pseudocode

```
// Pseudocode for Patient Registration
function registerPatient():
    prompt user for patient information
    create a new patient record
    populate record with user input
    save record to the patient database

// Pseudocode for Appointment Scheduling
function scheduleAppointment(patientID, date, time):
    check if the requested date and time are available
    if available:
        create a new appointment
        assign it to the patient
        update the appointment schedule

// Pseudocode for Billing
function generateInvoice(patientID, services):
    calculate total cost based on selected services
    create an invoice for the patient
    send the invoice to the billing department
```

Public Library Management Program

Objective

Develop a comprehensive library management program to enhance the efficiency of a public library's operations.

Project Description

Customer: Ms. Emily Anderson, the head librarian at the Greenfield Public Library.

Ms. Anderson has been struggling with manual record-keeping and inefficient check-in/check-out processes.

She envisions a digital solution to streamline library operations, improve user experience, and provide real-time information about available books.

Project Story

- Emily is passionate about making the library more accessible to the community.
- She envisions a system that can automatically catalog books, manage check-ins and check-outs, send overdue notifications, and provide an online catalog for patrons.
- Her dream is to create a library where anyone can easily find and borrow books, fostering a love for reading within the community.

Project Components

User Interface (UI) Design

- Create an intuitive user interface for both librarians and library patrons.
- Implement functionalities for book search, user registration, and check-in/check-out.

Transaction Management

- Design algorithms for smooth check-in and check-out processes.
- Handle reservation requests and book renewals.

User Management

- Enable user registration and login.
- Implement user roles (librarian and patron) with appropriate permissions.

Notification System

- Develop a notification system to send overdue reminders and other relevant messages to users.

Project Milestones

- Project kick-off and customer interview.
- UI design and database schema development.
- Book cataloging and transaction management.
- User management and notification system.
- Integration, testing, and bug fixing.
- User training and documentation.
- Final project presentation and handover to Ms. Anderson.

Stock Portfolio Tracker

Project Description

- Create a stock portfolio tracker to help investors monitor their investments.
- The application will allow users to add, view, and update stocks in their portfolio.
- Real-time stock price updates from an API will be integrated. This is a stock API I have used in a project.

```
API documentation https://www.alphavantage.co/documentation
500 requests per day free, 5 requests per minute
```

- Emphasis on user-friendly interface and data visualization.

Creative Customer Story

Meet John Smith: The Novice Investor

- John is a novice investor who recently started trading stocks.
- He's passionate about building his portfolio but struggles to keep track of his investments.
- John dreams of an easy-to-use application that can simplify his investment tracking process.
- Your team will play the role of developers tasked with creating a solution for John.

Project Milestones

- Project kickoff, team formation, and understanding user requirements.
- Designing the user interface for the stock portfolio tracker.
- Implementing the basic functionality to add and view stocks.
- Integrating real-time stock price updates.
- Adding functionality to update and delete stocks.
- Data visualization features for portfolio performance.
- Final testing, bug fixing, and project presentation.

Sample Pseudocode (General Structure)

```
# Define functions for the Stock Portfolio Tracker
function main():
    initialize_portfolio()  # Create an empty portfolio
    display_menu()          # Display the main menu
    while True:
        choice = get_user_choice()  # Get user's menu choice
        if choice == 1:
            add_stock()
        elif choice == 2:
            view_portfolio()
        elif choice == 3:
            update_stock()
        elif choice == 4:
            delete_stock()
        elif choice == 5:
            visualize_portfolio()
        elif choice == 6:
            save_portfolio()
        elif choice == 7:
            exit_program()

function add_stock():
    # Add a stock to the portfolio
    # Prompt user for stock details (symbol, quantity, purchase price, date)
    # Add stock to the portfolio data structure

# Implement other functions similarly

function visualize_portfolio():
    # Generate visual representation of portfolio performance
    # Options may include charts, graphs, or statistics

function save_portfolio():
    # Save the current portfolio data to a file

function exit_program():
    # Exit the application

# Call the main function to start the program
main()
```

Point of Sale System

Design and implement a Point of Sale (POS) system for a small business. The project will provide a practical application of programming concepts learned during the course. Students will work in teams to create a functional POS system and present their final product at the end of the semester.

Customer Story

- Invent a fictional customer who owns a small retail store.
- Describe the customer's business, its challenges, and why they need a POS system.
- Highlight the importance of the system for the customer's business operations.

Development Phase

- Identify key features such as inventory management, sales tracking, and user authentication.
- sample pseudocode to outline key algorithms and logic.
- Ensure data security and validation mechanisms.

Testing and Debugging

- Thoroughly test the system for functionality and user-friendliness.
- Debug and address any issues or errors.

More Project Ideas

Event Ticketing System: Develop a ticketing system for events, allowing users to purchase tickets online and event organizers to manage ticket sales.

Restaurant Reservation System: Develop a system for making and managing restaurant reservations, including table assignments and waitlist management.

Weather Forecasting App: Develop a weather forecasting application that provides detailed weather information and forecasts.

Recipe Recommendation System: Build a system that suggests recipes based on user preferences, dietary restrictions, and available ingredients.

Parking Reservation System: Design a platform that enables users to reserve parking spaces in advance, reducing congestion and improving convenience.

Fitness Center App: Design a fitness app that provides personalized workout routines and tracks users' progress. Track users, memberships, and accounting.

Language Learning Game: Develop an interactive game that helps users learn a new language through quizzes, vocabulary exercises, and challenges.

Blockchain-Based Voting System: Design a secure online voting system using blockchain technology.

Recipe Book: Build an app to store and search for recipes with ingredients and cooking instructions.

To-Do List Application: Develop a task management app with features like adding, editing, and deleting tasks.

Car Rental Management System: Create software for car rental companies to manage their fleet, reservations, and customer information.