Vanderwilt, Alexis

Trains

CSC 260 Object oriented design final project

# Table of Contents

[Problem Statement 2](#_Toc6759499)

[Problem statement 2](#_Toc6759500)

[Proposed solution 2](#_Toc6759501)

[Approximate timeframe 2](#_Toc6759502)

[Estimate of resources 2](#_Toc6759503)

[Approval Authority 2](#_Toc6759504)

[Program Requirements 2](#_Toc6759505)

[Trains V1.0 2](#_Toc6759506)

[Functional Requirements 2](#_Toc6759507)

[Non-functional Requirements 2](#_Toc6759508)

[Technical Design – Trains v 1.0 2](#_Toc6759509)

[UI Draft 3](#_Toc6759510)

[UI Trains V1 4](#_Toc6759511)

[Completeness & Appropriate Complexity 5](#_Toc6759512)

[Need to be able to view station map 5](#_Toc6759513)

[User can control the movement of the trains 5](#_Toc6759514)

[Users can control construction of tracks 5](#_Toc6759515)

[Packages spawn at random stations 5](#_Toc6759516)

[Users can earn money by delivering packages to correct station 5](#_Toc6759517)

[Can use money to build/buy trains and tracks and upgrade train storage 5](#_Toc6759518)

[Version Control 6](#_Toc6759519)

[UML Class Diagram 7](#_Toc6759520)

[Documented Test Cases & Testing 8](#_Toc6759521)

[Classes, Abstraction & Inheritance 8](#_Toc6759522)

[Encapsulation 8](#_Toc6759523)

[Polymorphism 9](#_Toc6759524)

[Unit Testing 10](#_Toc6759525)

# Problem Statement

Problem statement**:** Time management and the ability to keep track of multiple things at once is a crucial skill in today’s world. As a society, we are constantly juggling emails, texts, phone calls, meetings, and other important events. I am proposing a game that will help people practice this skill.

Proposed solution**:** Create a game where multiple trains are moving, and you must keep track of what train is going where, the capacity of each train, and make decisions about which packages need to be moved first.

Approximate timeframe**:** March 30th- April 20th

Estimate of resources**:** Developer time from one developer 5 hours a week for about 4 weeks, acquisition of developers’ tools.

Approval Authority**:** Jason Jenkins

# Program Requirements

Trains V1.0

Functional Requirements

* Need to be able to view station map
* User can control the movement of the trains
* Users can control construction of tracks
* Packages spawn at random stations
* Users can earn money by delivering packages to correct station
* Can use money to build/buy trains and tracks and upgrade train storage

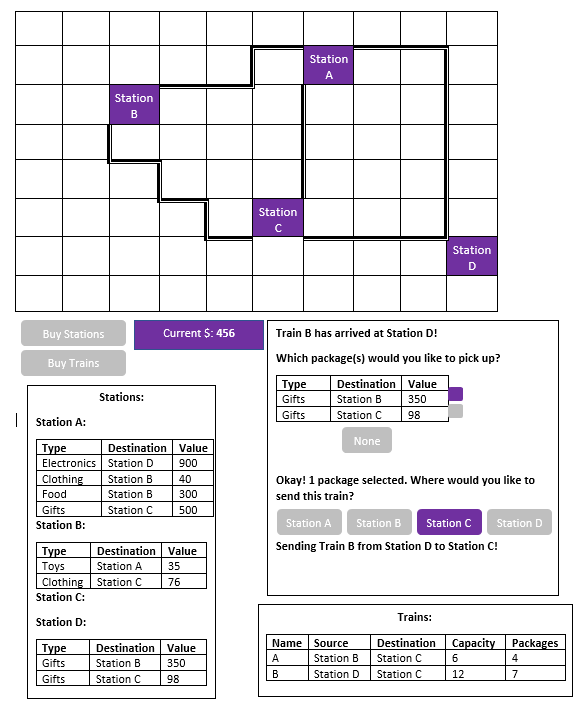
Non-functional Requirements

* Use console on the browser to keep track of the user’s activity

Technical Design – Trains v 1.0

* Version control system: GitHub
* Framework: ASP .NET Framework
* IDE: Visual Studio 2017
* Data Store: None
* Application Type: Windows Form

# UI Draft



# UI Trains V1

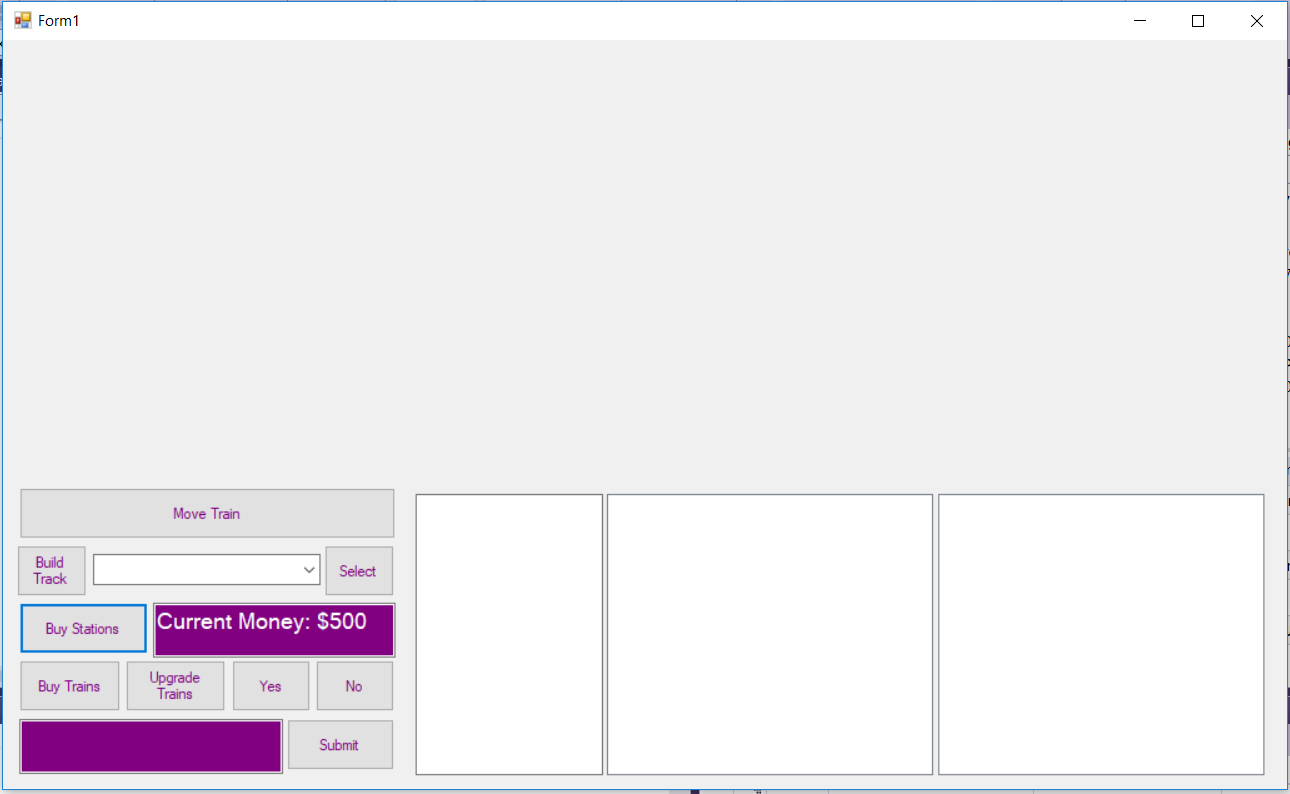


Figure Starting Design

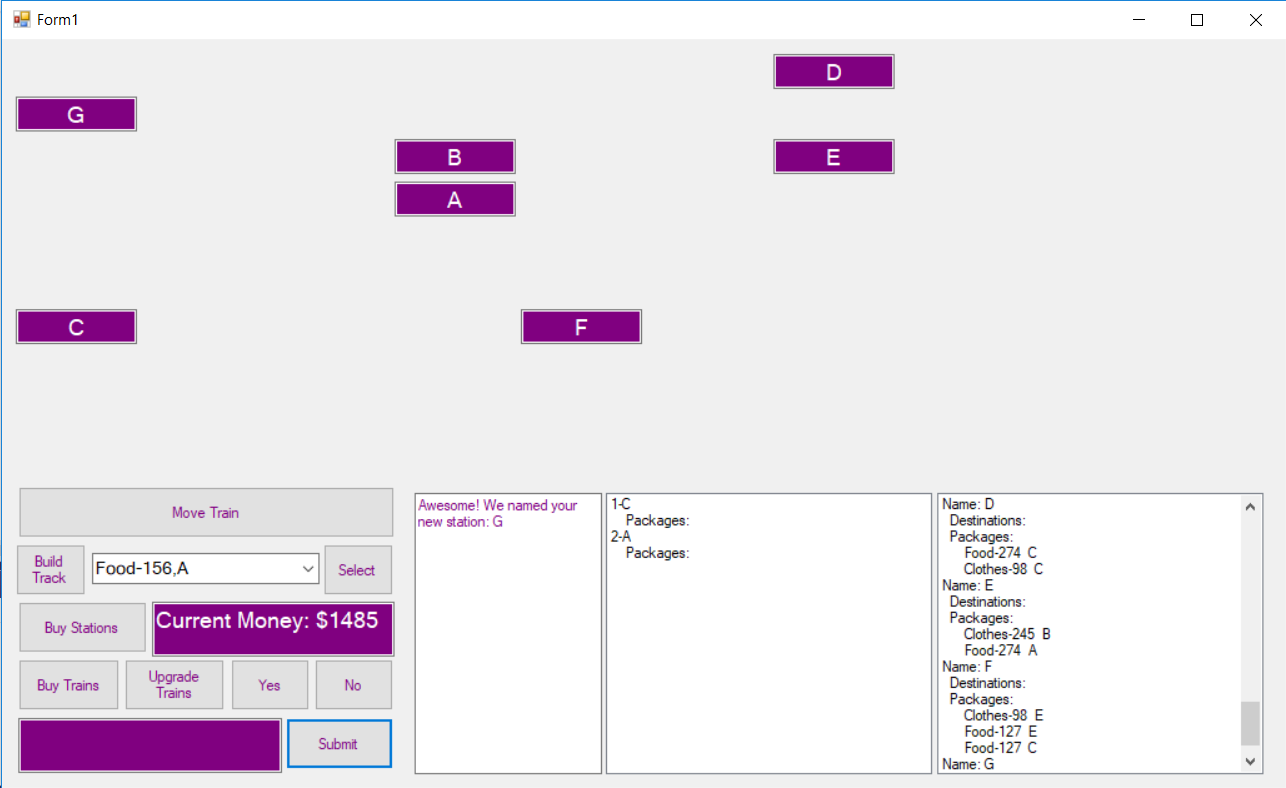


Figure After some purchases

# Completeness & Appropriate Complexity

Need to be able to view station map

This can be seen in [Figure 2](#_UI_Trains_V1). This was accomplished with a bunch of hidden text boxes that are hidden until they are purchased.

User can control the movement of the trains

The movement of trains is controlled by pressing the ‘move train’ button and then providing the name of the train that needs to move, followed by loading any packages that need to be moved also. Then, the destination is input and the train is moved.

Users can control construction of tracks

There is a button for building tracks. It asks for a source and destination station for the tracks. Tracks are only 1 direction, meaning if you have a track from Station A to Station B, you cannot move your train from Station B to Station A on this track.

Packages spawn at random stations

This is implemented through a ‘spawning Packages’ function that is called when certain transactions are made.

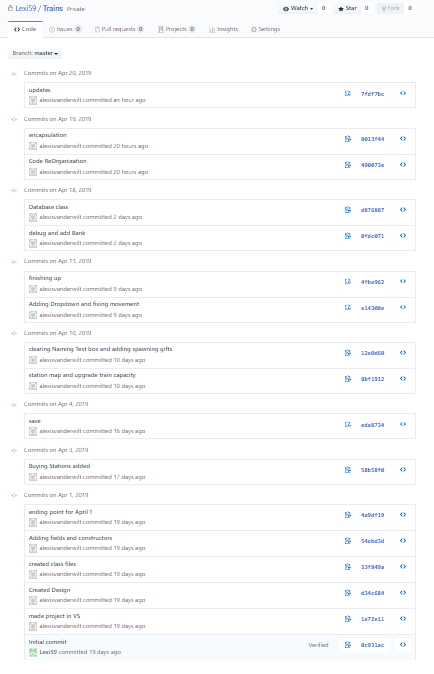
Users can earn money by delivering packages to correct station

In V1 Users can earn money by having packages on their train that need to be delivered to the station they are currently at. They do not have to manually deliver and/or unload the packages.

Can use money to build/buy trains and tracks and upgrade train storage

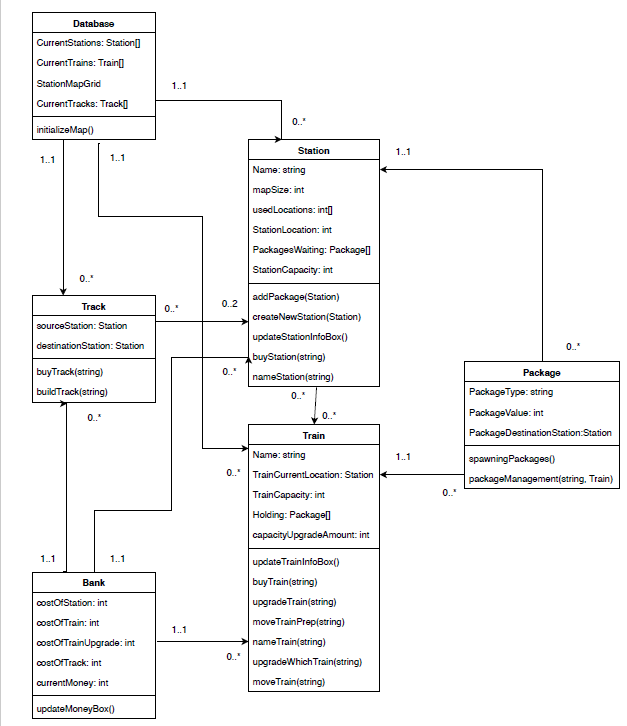
This works by having a Bank class that holds onto the current amount of money the user has and the cost of each type of item.

# Version Control



This github project has been shared with git user ‘mirageof hope’ with this invitation link: <https://github.com/Lexi59/Trains/invitations>

# UML Class Diagram



# Documented Test Cases & Testing

# Classes, Abstraction & Inheritance

I implemented 6 classes: Train, Track, Station, Package, Bank, Database. The Bank and Database classes were the most unusual. I implemented the Bank class to hold all of the prices for all of the items like Train, Track, Station, and Package. The Database held all the existing instances of the classes to make them accessible for everyone. I made this class an abstract class and let everyone inherit from it to allow everyone to easily make changes and access the data.

# Encapsulation

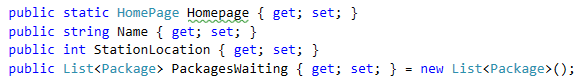
Encapsulation was used all throughout my project, but it is especially apparent in the StationClass.cs file. 

Figure Encapsulation Examples in StationClass.cs

# Polymorphism

I used polymorphism in the PackageClass.cs with the packageManagement method.

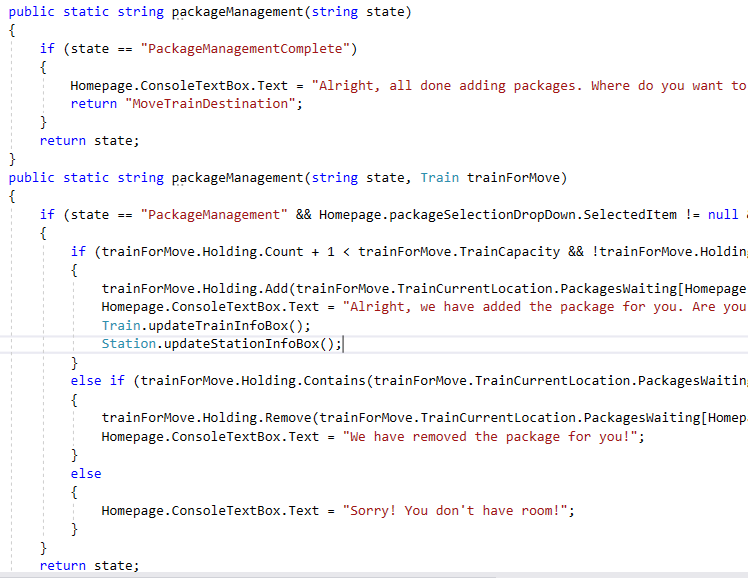


Figure polymorphism

# Unit Testing

I chose to implement Automated Unit testing. I built it with Nunit3 and the adapter for running it in the Visual Studio Enviornment. The file for my unit tests can be found in HomePageTests.cs in my project folder.

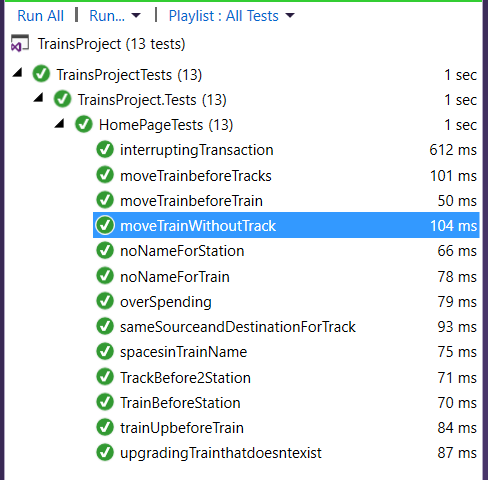


Figure Proof of Automated Unit Testing