

Using Software to Increase Efficiency In School Cafeterias

Western Governors University

001121729
Jeremiah McElroy

Table of Contents

Prompt A	4
Letter of Transmittal	4
Project Recommendation	5
Problem Summary	5
Application Benefits	5
Application Description	5
Objective and Hypothesis	6
Methodology	6
Funding Requirements	6
Stakeholders Impact	7
Data Precautions	7
Developer Expertise	7
Prompt B	8
Project Proposal	8
PROBLEM STATEMENT	8
CUSTOMER SUMMARY	8
EXISTING SYSTEM ANALYSIS	8
DATA	9
PROJECT METHODOLOGY	9
PROJECT OUTCOMES	10
IMPLEMENTATION PLAN	10
EVALUATION PLAN	11
RESOURCES AND COSTS	11
TIMELINE AND MILESTONES	11
Prompt C	13

Application Files	13
Prompt D	14
Post-implementation Report	14
Project purpose	14
Datasets	14
Data product code	15
Hypothesis verification	21
Effective visualizations and reporting	21
Accuracy analysis	22
Application testing	23
Appendices	24
Installation Guide	24
User Guide	24
Summation of Learning Experience	27
References	28

Prompt A

Letter of Transmittal

April 21th, 2020

Mr. Dave Snyder

Nevada R5 School System

920 N St

Kansas City, MO 99999

Mr. Snyder,

Currently the way lunches are kept track of and supplies for those lunches are ordered has several efficiency problems. Currently the lunches the students want are all taken by hand, manually put into an excel sheet, and printed off for the lunch staff. Due to the lack of data analysis on orders, the school generates unnecessary food wastage. It's imperative when running a school to save money and optimize tasks wherever possible due to a strict budget, and my program will be able to help you do that.

My program will solve this issue with the following solutions: First, instead of teachers writing down the amount of students wanting each lunch option and turning it into the office they will log into the application using their personal credentials and type in the students' choices. This data will then automatically be added to the database for future reference and automatically generate an order amount sheet for the cooks. On top of streamlining the process my program will also generate a report for the recommended amount of each food item which will reduce food wastages. The simplicity of this application allows it to be made on a tight budget; It will only cost \$6427 and take about three weeks. I will develop the program using Unity Engine and a SQLite file based database. The data used to construct this product is fabricated data generated with a realistic range, so there will be no sensitive information involved. None of this data used in this application will require extra legal or ethical precautions.

I fully expect this project to deliver on time with no major conflicts. I am experienced in working with unity applications, and working with SQL databases. I have constructed similar programs in the past and am confident in my timeframe and budget predictions.

Project Recommendation

Problem Summary

The way that lunches are currently kept track of and the way that supplies for those lunches are ordered has multiple efficiency issues. For one the students' lunch choices are written on paper by each teacher, turned into the office, and manually put into an excel sheet that is then printed off for the lunch staff. Also, due to the lack of data analysis on orders, the school produces higher wastage values than necessary.

Application Benefits

It's imperative when running a school to save money and optimize tasks wherever possible due to a strict budget. The data analysis my program will perform will help reduce wastage with only a small development, so it will be an investment for the school. My application will also reduce worker load and stress by optimizing the way lunch choice data is collected; Not only that, but it is crucial to optimize it in order to insure accurate data for the data analysis to work.

Application Description

The application is a Unity application that collects data from teachers' input about student lunch choices and stores it in an SQLite database; It then performs data analysis on the collected data to predict future lunch order amounts and generate the recommended order amount for each product and displays that information on a graph. It can also save the daily lunch choices to an excel sheet for the lunch staff.

Data Description

The school will collect the real data overtime, and the longer the program runs the more accurate it will be. For this example of the programs functionality we will be using randomly generated data based on possible trends. The data consists of the

date and amount of choices for an unlimited number of lunch choices.

Objective and Hypothesis

The objective of this program is to allow teachers to login and submit their students lunch choices for the day, collect that information, and perform data analysis to predict future food order amounts. The hypothesis is that the program will be able to predict student lunch choice data with 80% accuracy up to a week in advance, and 90% up to a month in advance.

Methodology

The method I will be taking in this project is the waterfall methodology. I've chosen the waterfall method because this project is small with clear set requirements, and all designs will have final approval before they are worked on. The waterfall methodology will follow six phases: Requirement Gathering, Design, Implementation, Verification, Deployment, Maintenance.

Funding Requirements

The program will run on the schools networked file system already in place so there will be no hardware costs for this project. The development cost will be \$6000 for 3 weeks of work \$50/hr for development plus tax making it \$6427. There are no expected long-term maintenance costs for this program.

Stakeholders Impact

This application will be a valuable tool for school administration to predict future order quantities which will help save money on losses as well as meet demand for students' lunch choices.

Data Precautions

There are no precautions needed for this project. There will be no sensitive data being used during testing, and there will be no actual cost numbers being used in the program, only lunch choice amounts and ingredients. I will use proper input validation to make sure no one can use an SQL injection or other basic attack; There is no need for PHP connections or encryption because there is no sensitive data, so I will use a direct SQLite connection.

Developer Expertise

I have worked on six different projects of a similar nature to this one. I have over three years of experience working in Unity Engine, and two working with SQL databases. There is no part of this project that is something I haven't done before, so I am extremely confident in my capability to deliver this product on time and on budget.

Prompt B

Project Proposal

PROBLEM STATEMENT

The way that lunches are currently kept track of and the way that supplies for those lunches are ordered has multiple efficiency issues. Each teacher writes their students' lunch choices on paper, turns them into the office, who then manually types them into an excel sheet and prints them off for the lunch staff. Also, due to the lack of data analysis on orders the amount of food wastage produced by the school.

CUSTOMER SUMMARY

The application is intended for use by teachers, lunch staff, office staff, and administration at the Nevada R5 School System. The purpose of the program is to allow teachers to login and submit student lunch choices instead of writing them on paper and bringing them to the office. It will also allow the office to print off the lunch choice amounts for the day for lunch staff instead of manually typing it into an excel sheet. The lunch staff and the administration will use the data generated by the program via machine learning to make food order amount decisions to reduce wastage. The customer will view the data generated via a graph with multiple view settings.

EXISTING SYSTEM ANALYSIS

Currently the school collects student lunch choices by having the teacher take each choice by paper, turn it into the office, then the office manually types the data into an excel sheet and prints it off for the lunch staff. Lunch order amounts are determined by lunch management based on intuition.

DATA

The data will be stored in a SQLite Database that is set up as a db3 file which Unity will have direct access to. During testing the data used will be fake test information. The database will store a table for user login information, the lunch choice options, students lunch choice information per day, lunch choice ingredients and recommended order amounts, orders, and meta data for the machine learning algorithm.

PROJECT METHODOLOGY

The method I will be taking in this project is the waterfall methodology. The steps are the following:

1. **Requirement Gathering:** I will meet with the Nevada R5 School System on two separate meetings. The first one will be to approve the business and deliverable requirements and the other will be to gather the UI requirements
2. **Design:** I will make database UML diagram, create class structure in c# that will hold database information. Design the UI within Unity and decide the object hierarchy within it
3. **Verification:** Use black-box testing methods to make sure the data works as intended
4. **Deployment:** The application will be installed on the local school computer
5. **Maintenance:** For the first day we will monitor the application to make

sure it works correctly. If there are any problems they will be fixed on the same day. Low budget and the simplicity of the project do not warrant any further maintenance

PROJECT OUTCOMES

Upon completion the program will be created on unity. It will be controlled via a graphical interface. The user will be able to log in, and have different buttons displayed based on whether they are an admin; Non-Admins have the option to add student lunch choice data, change their password, or save the days lunch choice spreadsheet. Admins will have access to add or modify users, add lunch choice options and products, and view the graph displayed past order amounts and future order amounts. The future order amounts will be predicted using the ML.net timeseries library. All the information will be stored on a db3 SQLite database file that will be installed on the schools main server.

IMPLEMENTATION PLAN

1. Create the class structure implementation in c# for the database
2. Create the functions to load the information from the database.
3. Create the UI
4. Connect the UI to the database implementation
5. Create the test data
6. Implement the machine learning model using the ml.net timeseries library
7. Export Final Build and test on different machines

EVALUATION PLAN

The project will pass requirements if it has a functioning UI delivering the functionality intended. The forecast model passes if it successfully stays under the lunch order amount for at least a week ahead in by day prediction mode.

RESOURCES AND COSTS

For this application, it will cost \$50/hr at an estimated 120 hours of development making it \$6000 plus tax making the total \$6427 for the budget. There are no other costs attributed to this project as the school already has the resources needed on hand.

TIMELINE AND MILESTONES

Development and deployment will take around 120 hours over the course of 3 weeks.

The milestones and timeline is provided in the chart below

CAPSTONE: TASK PIPELINE 20

Mile stone	Pre-requi si tes	Activity	Hours	Start	End
1	-	School Design and Goals Approval Meeting	5	5/3/21	5/3/21
2	1	Database Design and Development	16	5/3/21	5/5/21
3	1	School UI Design Approval Meeting	5	5/7/21	5/7/21

4	3	UI Design and Development	16	5/5/21	5/7/21
5	4	UI debug testing	8	5/8/21	5/8/21
6	2, 4	Implementing Database into Unity program	39	5/9/21	5/9/21
7	2	Implementing Machine Learning Algorithm	27	5/9/21	5/12/21
8	7,6,5	Debugging	8	5/22/21	5/22/21
9	8	Implement program in the School	6	5/28/21	5/28/21

CAPSTONE: TASK PIPELINE 21

Prompt C

Application Files

All unlisted files are part of the Engine or its dlls itself and not my project*

/schoollunchtracker

/Assets

/Database

This folder contains all the database files

Schoollunchtracker.db

schoollunchtracker.s3db

Schoollunchtracker.s3db-journal

/StreamingAssets

/Database

This folder contains a copy of the database. Unity uses the database folder to define the structure and the streaming assets file to fill it with data.

Schoollunchtracker.db

Secondary database file

Schoollunchtracker.s3db

This is the database file that is being used

Schoollunchtracker.s3db-journal

Secondary database file

/Graph

This contains the files for the graph display

/Images

This folder contains the images the graph uses

graph.prefab

Prefab the WindowGraph is attached to

WindowGraph.cs

Controls the Graph Display

menuBaseClass.cs

base class all menus inherit from

ChangePassword.cs

Controls the change password screen

DataBase.cs

Handles all methods accessing the database

LoginScreen.cs

Controls the login screen

MainMenu.cs

Controls the main menu screen

AddStudentLunchChoiceData.cs

Class used to add test data to the database

LuchDataScreen.cs

Controls the lunch data screen

ModifyLunchChoicesScreen.cs

Controls the modify lunch choices Screen

ModifyUsers.cs

Controls the change modify users screen

PredictFutureOrders.cs

Handels the ML.net machine learning

StudentLunchChoicesScreen.cs

Controls the student lunch choices screen/graph

Prompt D

Post-implementation Report

Project purpose

The purpose of this project is to create an application using Unity that collects student lunch choice data via teachers' input and stores it in an SQLite database; It then will perform data analysis on the collected data to predict future lunch order amounts and generate the recommended order amount for each product. This data will be accessible by saving the data to an excel sheet. It will also save the daily lunch choices to an excel sheet for the lunch staff.

CAPSTONE: TASK PIPELINE 23

Datasets

The test data for the program has three lunch choices, but only “chicken patty” will be used to evaluate this program's accuracy as it will work the same for any lunch choice. The data set consists of two values: date and order amount. The values are stored according to their lunchChoiceID and productID. If the view model(day, month, year) of the graph is changed within the program it will change the datasets output by avg all the values within the day, month, or year. An example of all three view modes' raw datas is shown below:

Day:

	Date	Amount of lunches	lunch_choice_id	lunch_choice_name
1	2019-08-24	297	1	Chicken Patty
2	2019-08-26	248	1	Chicken Patty
3	2019-08-27	241	1	Chicken Patty
4	2019-08-28	231	1	Chicken Patty
5	2019-08-29	242	1	Chicken Patty

Month:

	Date	Amount of lunches	lunch_choice_id	lunch_choice_name
1	2019-08	1526	1	Chicken Patty
2	2019-09	4905	1	Chicken Patty
3	2019-10	3211	1	Chicken Patty
4	2019-11	1818	1	Chicken Patty
5	2019-12	1504	1	Chicken Patty

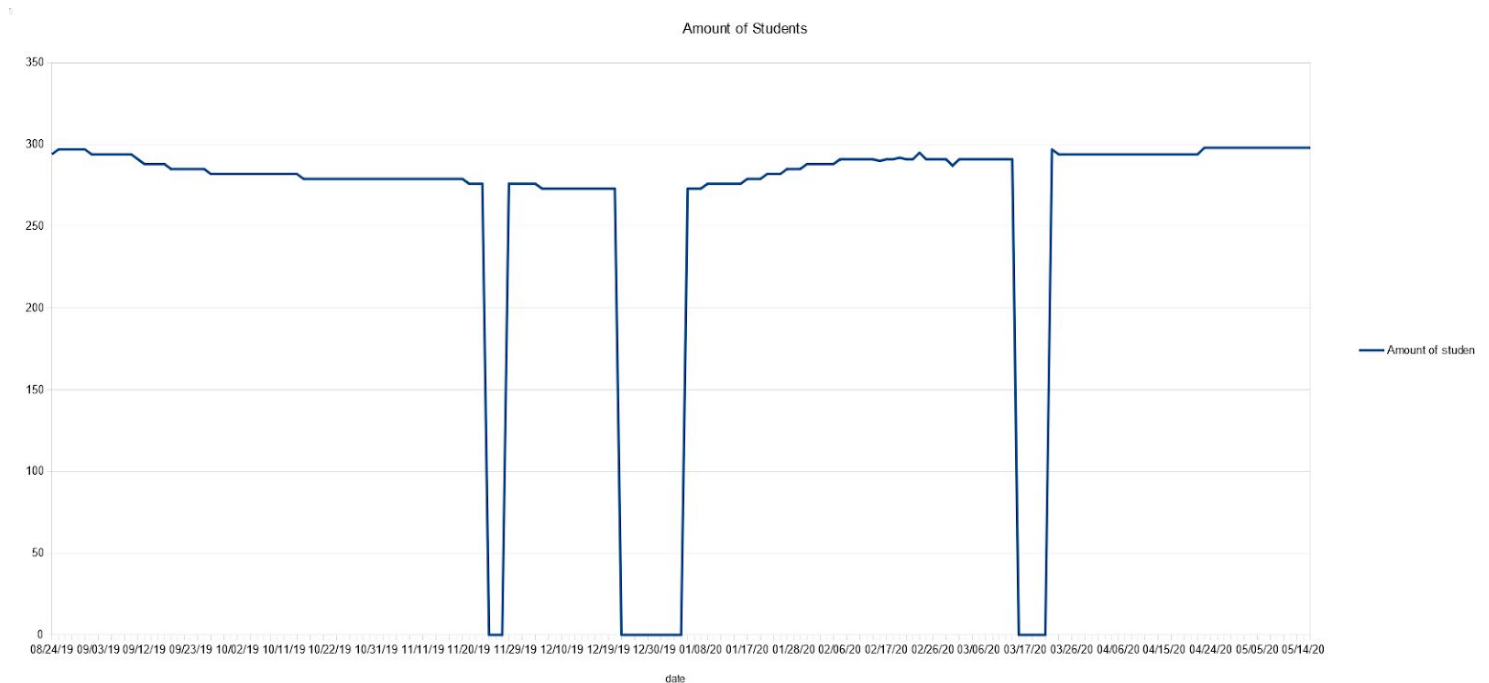
Year:

	Date	Amount of lunches	lunch_choice_id	lunch_choice_name
1	2019	13064	1	Chicken Patty
2	2020	10517	1	Chicken Patty

Data product code

First data was generated for a median amount of students in school for each day in a school year was generated by hand. I took into account holidays based on my local school districts holidays. I

made it lower in the winter months, and overall lower at the end of the year due to dropouts and transfers. Below is the final students in school graph:



Then, in the following code, I then calculated a random value from the median with a minimum and maximum value of 26% of the median. I used that random value to calculate a random value for each lunch choice in the test data and save them each to their own csv file. Below is the code used from the AddStudentLunchChoiceData.cs file:


```

private void GenerateRandomOrderAmounts(float percentageOfChoices, string outputFileName)
{
    string outputFilePath = "C:/Users/ghgfh/Downloads/Generated Data/Lunch Choices/" + outputFileName;
    List<Tuple<string, int>> studentsAtSchool = new List<Tuple<string, int>>();
    const float varianceOfChoices = 0.26f;

    using (StreamReader reader = new StreamReader("C:/Users/ghgfh/Downloads/Generated Data/ReducedData/Total Students Reduced.csv"))
    {
        string line;
        while ((line = reader.ReadLine()) != null)
        {
            var values = line.Split(',');
            string date = values[0];
            int orderAmount = int.Parse(values[1]);
            studentsAtSchool.Add(Tuple.Create(date, orderAmount));
        }
    }

    List<Tuple<string, int>> lunchChoiceAmounts = new List<Tuple<string, int>>();
    System.Random rand = new System.Random();
    foreach (var value in studentsAtSchool)
    {
        int amountOfLunchesMedian = Mathf.RoundToInt(value.Item2 * percentageOfChoices);
        int minimum = Mathf.RoundToInt(amountOfLunchesMedian - (varianceOfChoices * amountOfLunchesMedian));
        int maximum = Mathf.RoundToInt(amountOfLunchesMedian + (varianceOfChoices * amountOfLunchesMedian));
        int newRandomNumber = rand.Next(minimum, maximum);
        lunchChoiceAmounts.Add(Tuple.Create(value.Item1, newRandomNumber));
    }

    //write data to new csv
    using (StreamWriter writer = new StreamWriter(outputFilePath, false))
    {
        foreach (var value in lunchChoiceAmounts)
        {
            DateTime date = DateTime.ParseExact(value.Item1, "MM/dd/yy", System.Globalization.CultureInfo.InvariantCulture);
            writer.WriteLine(date.ToString("yyyy-MM-dd", System.Globalization.CultureInfo.InvariantCulture) + ", " + value.Item2);
        }
    }
}

```

The date and time is loaded into the program by accessing the SQL database and retrieving data from the proper queries. The query will dynamically change based on the display choice chosen.

Below is the code used to load data into the program. The code below is from the

AddStudentLunchChoiceData.cs file:

```

public List<Tuple<string, int>> GetOrderInformation(int selectedID, WindowGraph.DisplayMode displayMode, bool isProduct)
{
    List<Tuple<string, int>> studentChoices = new List<Tuple<string, int>>();

    //switches out the group by sql statement to group the sum of the days by the display mode
    string returnMode = "%Y-%m-%d";
    switch (displayMode)
    {
        case WindowGraph.DisplayMode.Day:
            returnMode = "%Y-%m-%d";
            break;
        case WindowGraph.DisplayMode.Month:
            returnMode = "%Y-%m";
            break;
        case WindowGraph.DisplayMode.Year:
            returnMode = "%Y";
            break;
    }

    dbconn.Open();
    dbcmd = dbconn.CreateCommand();

    //switch between grabbing product and lunchchoice information
    if (!isProduct)
    {
        dbcmd.CommandText = "SELECT * FROM (( SELECT strftime('" + returnMode + "', date), SUM(amount_of_lunches), lunch_choice_id " +
            "FROM studentlunchchoices WHERE lunch_choice_id = " + selectedID + " GROUP BY strftime('" + returnMode + "', date)) INNER JOIN ( SELECT " +
            "lunch_choice_id, lunch_choice_name FROM lunchchoices WHERE lunch_choice_id = " + selectedID +
            ") USING(lunch_choice_id));";
    }
    else
    {
        dbcmd.CommandText = "SELECT date, lunches, product_id FROM(SELECT * " +
            "FROM products INNER JOIN lunchchoices_products USING (product_id) " +
            "WHERE product_id = " + selectedID + ") INNER JOIN (SELECT strftime('" + returnMode + "', date) " +
            "date, lunch_choice_id, SUM(amount_of_lunches) lunches FROM studentlunchchoices " +
            "GROUP BY strftime('" + returnMode + "', date), lunch_choice_id) USING (lunch_choice_id) " +
            "INNER JOIN products USING(product_id)";
    }

    Debug.Log(dbcmd.CommandText);
    IDataReader reader = dbcmd.ExecuteReader();

    while (reader.Read())
    {
        string date = reader.GetString(0);
        int amount = reader.GetInt32(1);
        studentChoices.Add(Tuple.Create(date, amount));
    }

    dbconn.Close();
    dbcmd.Dispose();
    reader.Dispose();
    return studentChoices;
}

```

The date and time is then passed into the PredictFutureOrders file. In this function the data is then added to a ML.net Context object. The time series only outputs values, so the code keeps an index of the dates attached to it in a separate list:

```

public List<Tuple<string, int>> GetForecastedChoices(List<Tuple<string, int>> inputData, WindowGraph.DisplayMode displayMode)
{
    dateStrings = new List<string>();

    var inputDataAsTimeSeriesData = new List<TimeSeriesData>();

    foreach (Tuple<string, int> tuple in inputData)
    {
        if (displayMode != WindowGraph.DisplayMode.Year)
            inputDataAsTimeSeriesData.Add(new TimeSeriesData(DateTime.Parse(tuple.Item1), tuple.Item2));
        else
            inputDataAsTimeSeriesData.Add(new TimeSeriesData(DateTime.ParseExact(tuple.Item1, "yyyy", System.Globalization.CultureInfo.InvariantCulture), tuple.Item2));
    }

    if (inputDataAsTimeSeriesData.Count < 7)
    {
        return new List<Tuple<string, int>>();
    }
}

```

I use ML.net's ForecastBySsa function to perform time series analysis on the provided data and it outputs a forecast of the dates and choice amounts ahead based on the amount of data received. Here is the code performing this functionality:

```
MLContext context = new MLContext(); // defines a new ML context

//IDataView dataView = context.Data.LoadFromTextFile<ChoiceRecord>(path: dataPath, hasHeader: true, separatorChar: ',');
IDataView dataView = context.Data.LoadFromEnumerable(inputDataAsTimeSeriesData);

List<Tuple<string, int>> newTuples = new List<Tuple<string, int>>();

// build a training pipeline for forecasting data
var pipeline = context.Forecasting.ForecastBySsa(
    "Forecast",
    nameof(TimeSeriesData.amountOfChoices),
    windowSize: (inputDataAsTimeSeriesData.Count > 365) ? 365 : inputDataAsTimeSeriesData.Count / 4,
    seriesLength: inputDataAsTimeSeriesData.Count + 1,
    trainSize: ((inputDataAsTimeSeriesData.Count > 365) ? 365 : inputDataAsTimeSeriesData.Count / 4) * 2 + 1,
    horizon: (inputDataAsTimeSeriesData.Count > 365) ? 365 : inputDataAsTimeSeriesData.Count / 4);

// train the model
var model = pipeline.Fit(dataView);

var forecastingEngine = model.CreateTimeSeriesEngine<TimeSeriesData, TimeSeriesForecast>(context);

var forecasts = forecastingEngine.Predict();
```

The output data is then cleaned by using the date index to determine the outputs starting date, then assigns incremented dates to each date thereafter. It is then output back to the graph. You can see the code for this below:

```

int index = 0;
DateTime currentDate = inputDataAsTimeSeriesData.Last().date;

//make sure next prediction date starts in the school year
int schoolStartMonth = 8;
int schoolStartDay = 24;
int schoolEndMonth = 5;
int schoolEndDay = 15;

if (displayMode == WindowGraph.DisplayMode.Day)
{
    currentDate.AddDays(1); // stops it from forecasting current day
    //makes sure date falls within school year
    if ((currentDate.Month < schoolStartMonth && currentDate.Month > schoolEndMonth) ^
        (currentDate.Month == schoolStartMonth && currentDate.Day < schoolStartDay) ^
        (currentDate.Month == schoolEndMonth && currentDate.Day >= schoolEndDay))
    {
        currentDate = new DateTime(currentDate.AddYears(1).Year, schoolStartMonth, schoolStartDay);
    }
}
else if (displayMode == WindowGraph.DisplayMode.Month)
{
    currentDate = currentDate.AddMonths(1); // stops it from forecasting current day
    //makes sure date falls within school year
    if ((currentDate.Month <= schoolStartMonth && currentDate.Month >= schoolEndMonth))
    {
        currentDate = new DateTime(currentDate.AddYears(1).Year, schoolStartMonth - 1, 1);
    }
}

foreach (var forecast in forecasts.Forecast)
{
    string currentDateString;

    //increments day properly
    if (displayMode == WindowGraph.DisplayMode.Day)
    {
        currentDate = currentDate.AddDays(1);
    }
    else if (displayMode == WindowGraph.DisplayMode.Month)
    {
        currentDate = currentDate.AddMonths(1);
    }
    else
    {
        currentDate = currentDate.AddYears(1);
    }
    currentDateString = currentDate.ToString("yyyy-MM-dd");
    //Debug.Log(currentDateString + ", " + Mathf.RoundToInt(forecast));
    newTuples.Add(Tuple.Create(currentDateString, Mathf.RoundToInt(forecast)));

    index++;
}

return newTuples;
}

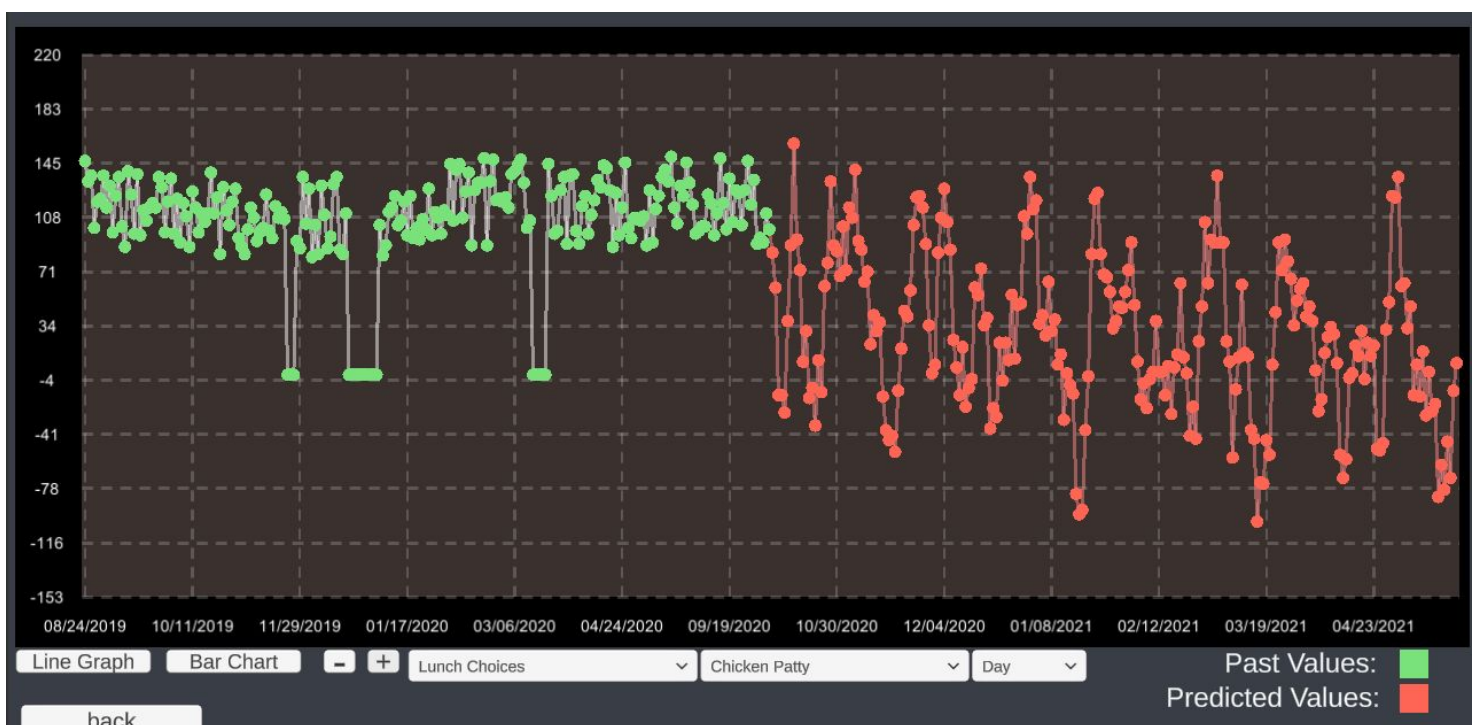
```


Hypothesis verification

The hypothesis is that the program will be able to predict lunch choice data with 80% accuracy upto a week in advance, and 90% upto a month in advance. The hypothesis cannot be completely confirmed until more real data is collected, as the more data we have the more accurate the program will be. Based on the test data the program passed at predicting data upto a month in advance at 86%, and failed for a week in advance at 33% when the program is in the day setting. If the program follows the same trend when more data is collected it will be better for predicting future months total sales.

Effective visualizations and reporting

The data can be viewed in the graph within the program. The program can display as a bar graph or a line graph and you can switch between the lunch choice. You can also show the associated products and their predictions, but they are just an extra feature and not accurate. The program shows the predictions upto a year in advance. Below is the data for the chicken patty lunch choice:





Accuracy analysis

The total lunch choice amount for each week and month was compared to the generated datas last school year. The accuracy of each week and month was then averaged. The program is considered acceptable if it can predict either a week ahead or a year ahead at least 80% in one category on the “day” setting. For a week the data points are too large within the document, but the average was only 33% and is a fail. For a month it passed at 86% and the data points are listed below:

Month	Accuracy
Aug	82%
Sept	97%
Oct	78%
Nov	90%
Dec	84%
Jan	88%
Feb	87%
March	77%
April	93%
May	87%
Avg Accuracy:	86%

Application testing

Acceptance testing was used to make sure the monthly or weekly accuracy percentage met the requirement as detailed above. Black-box testing, where the user doesn't know the inner workings of the program, was performed to make sure the UI functioned properly. Three random people were chosen to use the program with instructions on how to do so with the goal to make the program break.

Appendices

Installation Guide

Prerequisites:

Microsoft .NET Framework 2.8 (you most likely already have this installed)

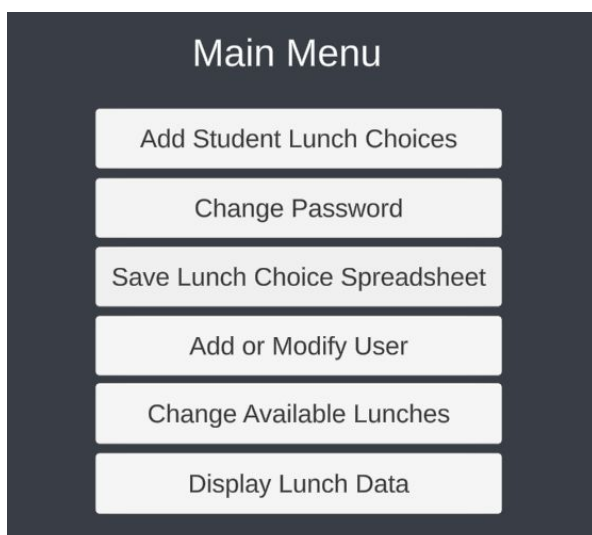
Instructions:

- 1) Unzip the program.
- 2) Drag the folder “64” to whatever folder you want.
- 3) Launch the application by double clicking: “School Lunch Tracker.exe”

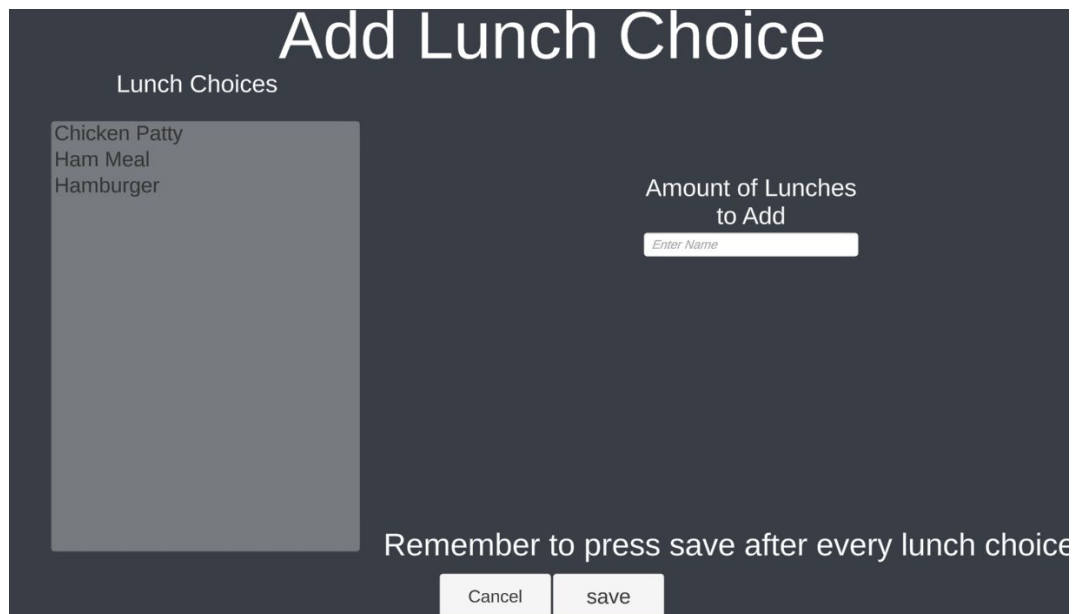
Clear the test data by accessing the database located at “StreamingAssets/Schoollunchtracker.s3db” with your program of choice

User Guide

When the program launches login with the default username and password: “admin”, and “password”. The main menu will then pop up as seen below. Only admins can see the last three buttons.

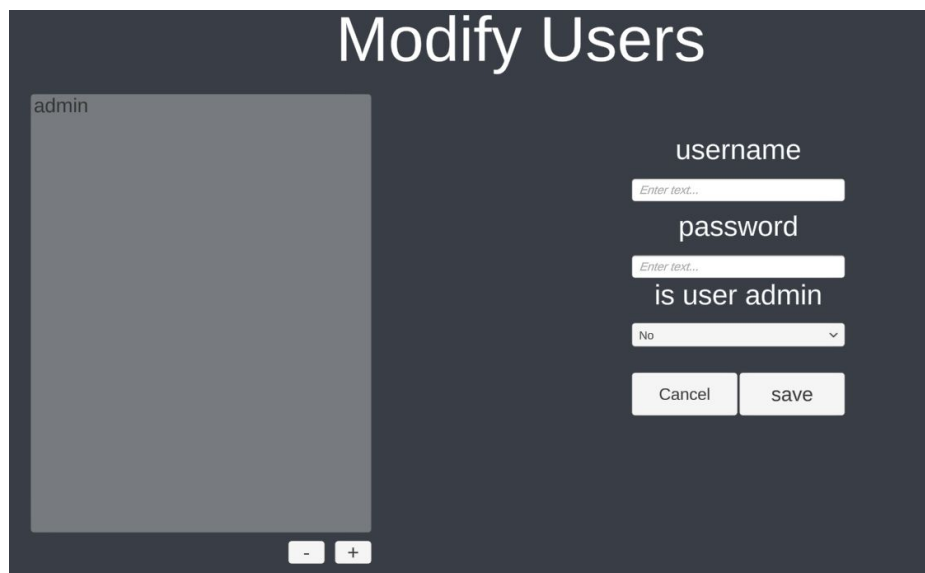


The first button opens up the Add Student Lunch Choices Screen as shown below. Choose the lunch choice on the left and type in the amount of lunches to add for today. Remember to click save after every lunch.



The 'Add Lunch Choice' screen has a dark blue background. At the top, the title 'Add Lunch Choice' is in large white font. Below it, on the left, is a box titled 'Lunch Choices' containing a list: 'Chicken Patty', 'Ham Meal', and 'Hamburger'. To the right of this list is a text input field labeled 'Amount of Lunches to Add' with a placeholder 'Enter Name'. Below the input field is a reminder text: 'Remember to press save after every lunch choice'. At the bottom are two buttons: 'Cancel' and 'save'.

The second button main menu button allows you to change your password, it will only allow alphanumeric characters. The third menu button will open up a prompt and ask you where to save the lunch choices for today as a csv file, choose a folder or an existing file. The fourth button will open up the modify users screen as shown below:



The 'Modify Users' screen has a dark blue background. At the top, the title 'Modify Users' is in large white font. On the left is a list box containing the name 'admin'. To the right of the list box are four input fields: 'username' (placeholder 'Enter text...'), 'password' (placeholder 'Enter text...'), 'is user admin' (placeholder 'No'), and a dropdown menu. At the bottom are two buttons: 'Cancel' and 'save'. Below the list box are two small buttons: '-' and '+'.

To add a new user press the “+” button at the bottom left. To remove a user select a user and click the “-” button. Pressing save will update a user with the values stored in the entries on the right. The fourth main menu button opens up the “modify lunch options” screen as shown below:

Modify Lunch Options

Chicken Patty Hamburger Ham Meal	Mashed Potatoes Frozen Chicken Patty Roll Milk French Fries Pickels hamburger buns hamburger patty Ham	
<input type="text" value="Enter Name"/> - +	<input type="text" value="Enter Name"/> - +	<div>+ -</div>

back

The available lunch choices are shown on the left, the available products in the middle, and the products associated with the lunch choice on the right. To add a product or lunch choice type name into the input below and click the (+) below. To add an associated product to a lunch choice click the selected lunch choice, then the selected product from the middle, then click the “+” button between the two right columns. To remove a product choose a lunch choice, a product on the left column, then click the “-” button between the two right columns. The last column will display the student lunch choice data screen as shown below:



Click the “Line Graph” and “Bar Chart” buttons at the bottom left to display a line graph and a bar chart respectively. The “-” and “+” will increase and decrease the amount of values being shown. The first drop down from the left will allow you to change whether to view the lunch choices or the associated products. The second will allow you to switch between the lunch choices and products, the last will allow you to view by day, month or year. The prediction is only accurate in the day setting.

Summation of Learning Experience

Knowledge from previous projects greatly aided me in my ability to complete this project.

However, I still had a lot to learn throughout this project. I had to learn what time series analysis is and how to use it with ml.net. I learned how to add external libraries that are not Unity specific to unity; In fact this probably took up the most amount of time as I had to figure it out myself, google had no answers. I also had to learn how to set up an SQLite database as while I’ve used many SQL databases, I’ve never created one. This project was extremely challenging and had many setbacks every step of the way, but I was able to overcome the challenges of forcing different dlls to work together that probably shouldn’t.

References

Farragher, M. (2019, November 19). Predict sales spikes with c# and ml.net machine learning. Retrieved February 09, 2021, from <https://medium.com/machinelearningadvantage/predict-sales-spikes-with-c-and-ml-net-machine-learning-643bbc8af835>

Brownlee, J. (2020, August 14). Time series forecasting as supervised learning. Retrieved February 09, 2021, from <https://machinelearningmastery.com/time-series-forecasting-supervised-learning/>

Metwalli, S. (2020, September 21). How to choose the right machine learning algorithm for your application. Retrieved February 09, 2021, from <https://towardsdatascience.com/how-to-choose-the-right-machine-learning-algorithm-for-your-application-1e36c32400b9>