COSC345 ASSIGNMENT 1

Team ATAK:

Andrew Booth	ID: 8539924
Tommy Light	ID: 1589489
Andrew Goh	ID: 3457439
Kris Mao	ID: 7248988

Datasets:

We are using two main datasets along with an extended dataset.

1. The Movies Dataset
2. Food dataset
3. Extended IMDB movies dataset (Only using the csv files)

The first dataset contains metadata of over 45000 movies up to the year 2017. It includes essential movie information such as title, overview, ratings, language and genres.

The second dataset is a recipe dataset that includes 2 million recipes of a variety of snacks and meals. The dataset also includes recipe instructions and a Named Entity Recognition column which will be useful for our recommendation algorithm.

The extended dataset contains more recent movies where I selected only movies from the year 2018 and onwards to add more movie information to the first dataset. Although it was someone's github repository. I asked permission to use his csv files which he agreed to.

The two datasets go together as they each have a column that classifies the rows together. The combined movie dataset would have a genre column which can be used for our recommendation algorithm. The food dataset has an Named Entity Recognition where its values can be paired with a certain genre. The combination of multiple movie genres and food keywords would allow us to create a wide range of movie and food recommendations for the user.

What are we building:

We are building a movie recommendation application that recommends the user food to pair with the movie.

The application will have two modes available to the user.
The first mode is a "Conditional" mode where the user can set 2 conditions:

- Genre: The user can pick up to 2 genres, a random option is available
- Rating: The user can pick a rating up to 10. The movie selected would be of that rating or near that rating

Both options can be set to a random value.
A movie that satisfies those conditions would have its information displayed.

The second mode would be a "Knockout" mode.

The user selects the number of "rounds" from 1 - 5 and the genre of the movie, both options can be set to a random value. The application would present two movies with their details and the user selects one out of the two. This repeats until the number of rounds are completed and the chosen movie will be displayed.

A food recommendation menu will pop out and give 3 options to the user after each of the movie selection modes are completed. The user can select one of the 3 options and further details about the recipe will be displayed. The random option is also available for the 3 options.

How are we building it:

We will build the application using C++. We will be using Visual Studio Community as our main IDE for running the application code. We chose to use Visual Studio as not only does it provide better support for C++ and its additional libraries than Visual Studio Code but more importantly, it has support for VCPKG which is used to install additional C++ libraries easily. We will be using CMAKE to build our application which allows anyone to be able to build the application as long as they have the required files and CMAKE settings.

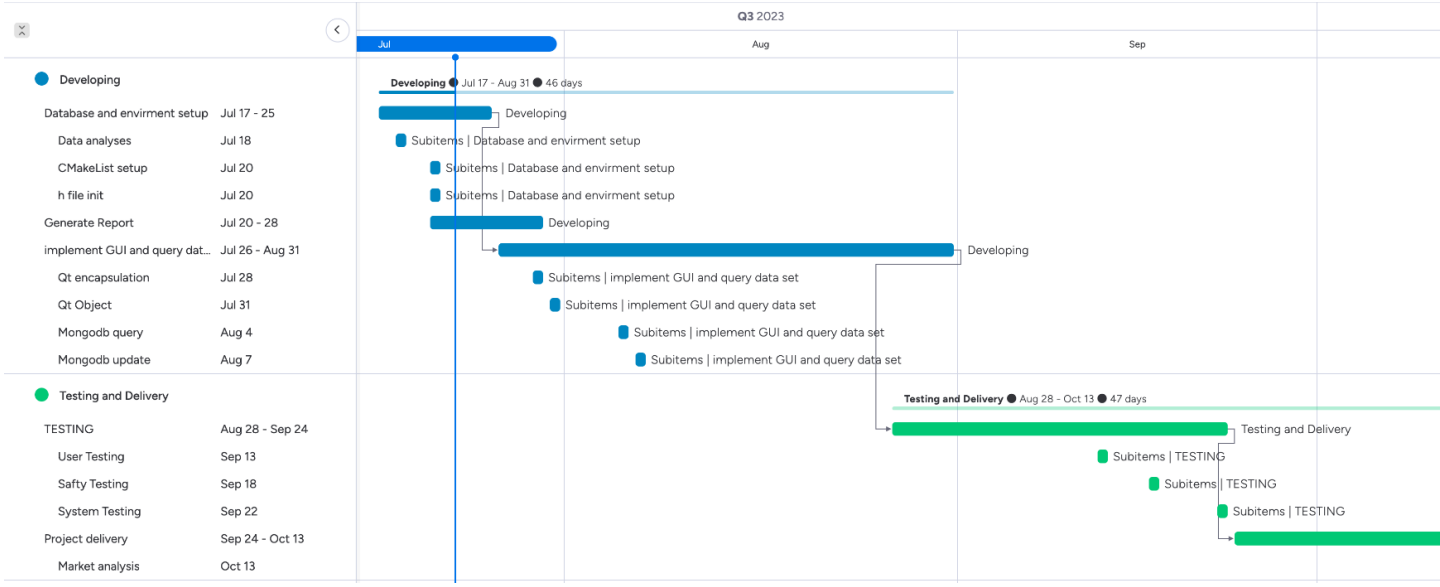
Unfortunately for our group member Kris, MACOS has rather bad support for Visual Studio so he has to use Visual Studio Code instead. He would have to manually configure the directories to include the additional C++ libraries instead of it being automatically configured by CMAKE and VCPKG.

We will be using additional C++ libraries such as Mongo-cxx and QT5. Mongo-cxx is a MongoDB C++ driver which will allow us to access our movie and food data that is stored in MongoDB. Any data query functions will be available through the MongoDB library.

For our GUI interface, we will be using the QT library for development. We will be using QT5 as it is widely accessible and has enough support for it.

Gantt Chart:

The diagram below is a snapshot of our Gantt chart. Essentially, we aim to finish developing our alpha version of the application near the end of August and start our testing immediately after the alpha version is finished.



What existing applications are similar:

We explored other recommendation websites/applications with similar purposes to gauge where our application would stand. We sat down as a group and used each application. We noted down their advantages and disadvantages in the tables below.

Website: PickAMovieForMe	
Advantages	**Disadvantages**
Large collection of movies.	Doesn't allow free searching of movies.
Mood/occasion are considered when providing a recommendation.	Lengthy selection process. (No random button)
Can watch trailers directly on their website.	No food recommendations.
New movies are added consistently.	

Website: MovieLens	
Advantages	**Disadvantages**
Custom tags can be applied to each movie.	Require login to use.
Large collection of movies.	Really really slow loading times.
Up to date.	Outdated interface.
Easy to use.	

Website: Moodie Foodie	
Advantages	**Disadvantages**
Use of machine learning.	Must register.
Recommends food as well as delivers it.	Must download and install required packages.
	Users must have a background in coding (python).
	Locally hosted.
	Odd usage of words

Website: ***Google web browser food recommendation***	
Advantages	**Disadvantages**
Easy to use	No movies recommended with food
Highly commercial	Require a large database for recommendations
Rating system	
Review base on users	

Evidence of customer interest in application:

From browsing the above applications, we believe that we found evidence of customer interests for our application. In MovieLens, there are many custom tags found with the movies on the website which suggests a large number of users. There are also many applications for food recommendation such as the provided examples above. However, there are none that recommends both movies and food together which will make our application unique.

Risk Analysis:

In the following risk analysis we rated each risk's likelihood on a scale from 0-1 to indicate the probability of its occurrence. We also rated the impact each risk would have on the project by assigning it a value that indicates the number of days of work the risk could add. We also calculate the exposure which is a function of the likelihood and impact to compare each risk.

Risk Type	Risk	Likelihood	Impact	Exposure	Plan/Action
Staff	Sick/absent	0.8	4	3.2	Since the time for the project is fixed we will control the scope of the scale to allow for the time that people are unavailable.
System	Crashes/has bugs	1	2	2.0	Bugs and system crashes happen often while developing a program. To fix those kind problems will take 2 days.
System	System been hacked	0.1	9	0.9	Since this App most time will run locally so most of the time it is safe. If a hacking incident happens, we can just shut it down to protect our App.
Project	Using agile methodology for the first time	0.4	8	3.2	Members with experience will share that

					experience and try to teach the other members how to follow the agile methodology.
System	The system not meeting the end-user needs or wants.	0.3	10	3	Involve the end-user in the production of the system.
Hardware	Blackout	0.1	10	1	There is nothing we can do, except praying for the power comes back sooner.
Staff	Lack of experience using C++ and Visual Studio.	0.3	4	1.2	Experienced members will share their experience with using the language/IDE and members without experience will spend some extra time learning them.
Staff	No experience integrating a MongoDB connection into a C++ application	0.8	5	4	We will control the scope of the project to allow us the additional time required to learn how to create an app with this connection.

Mock Up:

The image below shows our mock up for the application's menu. There would be a menu at the left panel where the two available modes can be selected despite the image showing 4 dashboards. A random list of movies is displayed on the right to give a user an immediate chance to view the movie's details if one catches their attention.