GOSPOIL

By Cody and Kaito

WHAT ARE OUR GOALS?

Our main goal is to provide a movie/series information app which provides more information compared to our other competitors .
We intend to provide an easy to access app which can be easily used by anyone to quickly find information on a film/series.

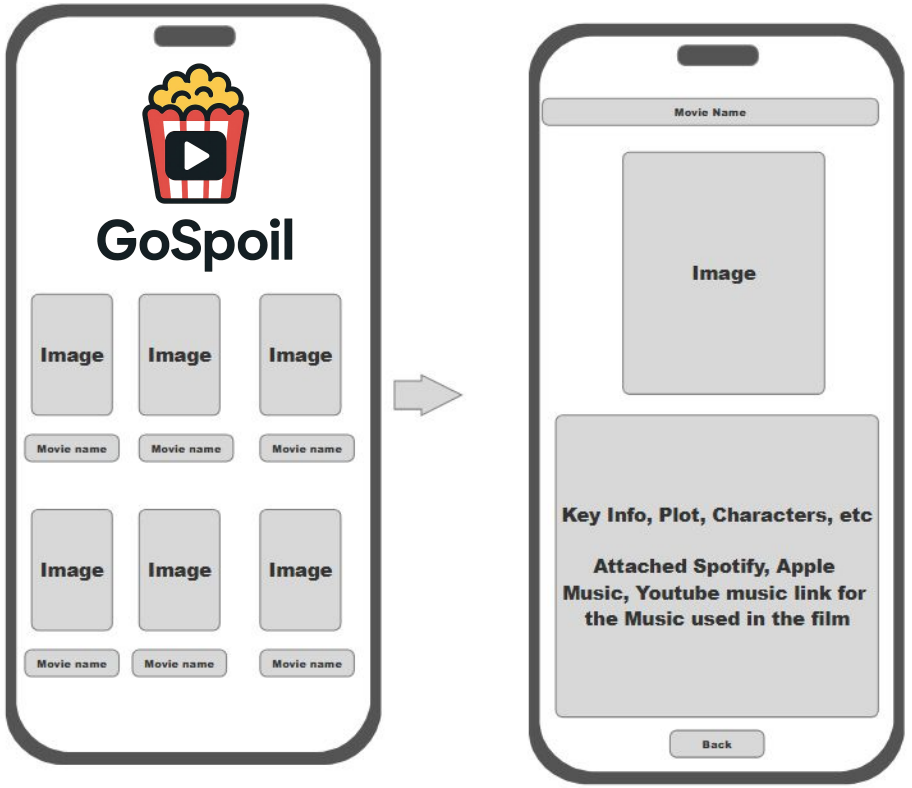
GANTT CHART

Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9
Proposal									
Create and distribute survery									
Identify FR, NFR and Constraints									
Determine Scope									
Generate alternate designs									
Write evaluation critea									
Select Preferred Design									
Create preferred design									
Identify ethical and legal issues									
Research object detection									
Research image processing									
Data collection and labelling									
Train the model									
Test the model									
Develop the project									
Test the project									
Collect data									
Evaluate the solution									

WIRE FRAME

We plan for the app to provide...

- Key information about the film/show
- Provide a link for the music used in the film/show
- Key Characters info
- Quotes
- Filming locations
- A list of actors
- A feature for small directors to post and advertise their films (TBD)



PROGRAM PROTOTYPE

Currently the gospoil app is able to locate a wide range of movies and provide a summary for each and everyone. It is also able to give the key characters in the film, and the directors of the film, while also giving an average rating of the movie. Unfortunately we were unable to integrate a link for all songs in the movie since it was blocked by a paywall.

The Gospoil app is connected to the OMBD database which is a database which has information on movie, movie posters, etc. OMBD is used to display the image posters while also providing a average rating, and giving a wide range of movies to search.

PROGRAM PROTOTYPE: Summary System

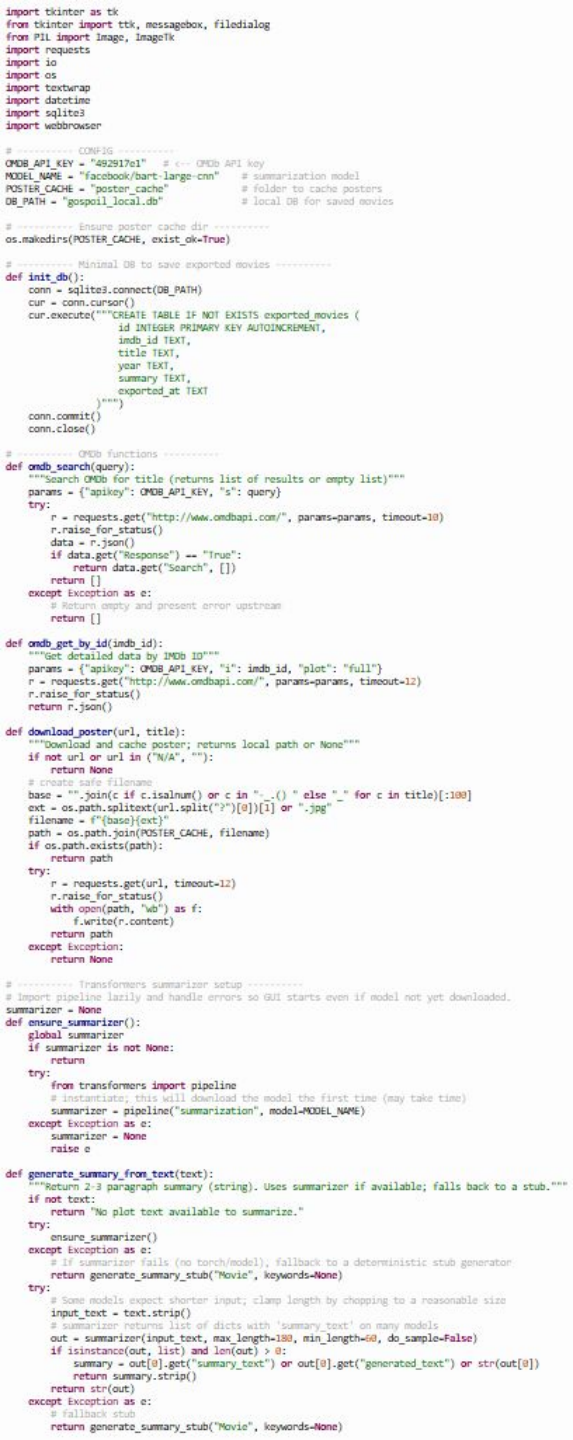
The GoSpoil app is connected to an Ai called Hugging Face Ai (Text generation model). This Ai helps to generate a summary, when the summary button is clicked. The Ai generates a 1-2 sentence summary of the movie when you click the quick spoil button and when you click the generate summary button the Ai is able to generate a summary up to 5 sentences long. After the Ai summary, you can click the Export Txt button which will prompt you to download a text file which consists of the movie title, When the movie was released, The Ai summary of the movie, The IMDb id (Can use it to search the movie on IMdb), and it also have the time and date that you exported the file,

```
Title: A Minecraft Movie
Year: 2025

Summary:
Four misfits are suddenly pulled through a mysterious portal into a bizarre cubic wonderland that thrives on imagination. To get back home they'll have to master this world
while embarking on a quest with an unexpected expert craf...

Metadata:
IMDb ID: tt3566834
Exported at: 2025-10-06T10:47:17.625230Z
```

PROGRAM PROTOTYPE: Problems Faced

The problems that were faced during the creation of the GoSpoil app is the movie song finder being blocked by the paywall. Due to this we decided to not incorporate it into the GoSpoil app, but it still may be incorporated in the future if we find a free option. Another problem that we faced was that we needed to add a lot of packages to Thonny such as Hugging face hub, Tkinker (allows you to open a application and have displays, etc) and much more. We had trouble adding them into thonny as we could not download the single package, so we asked Chatgpt what would be a fix and we ended up having to download the packages of the packages when ended up working. Another problem which we faced was the amount of errors that came up when making the code. With the help of ChatGPT guiding us and the Thonny Debug system we were able to spot out the errors and managed to fix them. Another problem is that due to the OMBD integration, The movie list is unable to load while connected to the school internet. As of right now we do not know why it does not load. A solution for this may be a Mobile phone hotspot but this is not confirmed.

Not all the code (Couldn't fit 400 lines into 1 screenshot)



```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from PIL import Image, ImageTk
import requests
import io
import os
import textwrap
import datetime
import sqlite3
import webbrowser

# ---------- CONFIG ----------
OMDB_API_KEY = "492917e1"  # <-- OMDb API key
MODEL_NAME = "facebook/bart-large-cnn"    # summarization model
POSTER_CACHE = "poster_cache"             # folder to cache posters
DB_PATH = "gospoil_local.db"              # local DB for saved movies

# ---------- Ensure poster cache dir ----------
os.makedirs(POSTER_CACHE, exist_ok=True)

# ---------- Minimal DB to save exported movies ----------
def init_db():
    conn = sqlite3.connect(DB_PATH)
    cur = conn.cursor()
    cur.execute("""CREATE TABLE IF NOT EXISTS exported_movies (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    imdb_id TEXT,
                    title TEXT,
                    year TEXT,
                    summary TEXT,
                    exported_at TEXT
                )""")
    conn.commit()
    conn.close()

# ---------- OMDb functions ----------
def omdb_search(query):
    """Search OMDb for title (returns list of results or empty list)"""
    params = {"apikey": OMDB_API_KEY, "s": query}
    try:
        r = requests.get("http://www.omdbapi.com/", params=params, timeout=10)
        r.raise_for_status()
        data = r.json()
        if data.get("Response") == "True":
            return data.get("Search", [])
        return []
    except Exception as e:
        # Return empty and present error upstream
        return []

def omdb_get_by_id(imdb_id):
    """Get detailed data by IMDb ID"""
    params = {"apikey": OMDB_API_KEY, "i": imdb_id, "plot": "full"}
    r = requests.get("http://www.omdbapi.com/", params=params, timeout=12)
    r.raise_for_status()
    return r.json()

def download_poster(url, title):
    """Download and cache poster; returns local path or None"""
    if not url or url in ("N/A", ""):
        return None
    # create safe filename
    base = "".join(c if c.isalnum() or c in "._-() " else "_" for c in title)[:100]
    ext = os.path.splitext(url.split("?")[0])[1] or ".jpg"
    filename = f"{base}{ext}"
    path = os.path.join(POSTER_CACHE, filename)
    if os.path.exists(path):
        return path
    try:
        r = requests.get(url, timeout=12)
        r.raise_for_status()
        with open(path, "wb") as f:
            f.write(r.content)
        return path
    except Exception:
        return None

# ---------- Transformers summarizer setup ----------
# Import pipeline lazily and handle errors so GUI starts even if model not yet downloaded.
summarizer = None
def ensure_summarizer():
    global summarizer
    if summarizer is not None:
        return
    try:
        from transformers import pipeline
        # instantiate; this will download the model the first time (may take time)
        summarizer = pipeline("summarization", model=MODEL_NAME)
    except Exception as e:
        summarizer = None
        raise e

def generate_summary_from_text(text):
    """Return 2-3 paragraph summary (string). Uses summarizer if available; falls back to a stub."""
    if not text:
        return "No plot text available to summarize."
    try:
        ensure_summarizer()
    except Exception as e:
        # If summarizer fails (no torch/model), fallback to a deterministic stub generator
        return generate_summary_stub("Movie", keywords=None)
    try:
        # Some models expect shorter input; clamp length by chopping to a reasonable size
        input_text = text.strip()
        # summarizer returns list of dicts with 'summary_text' on many models
        out = summarizer(input_text, max_length=180, min_length=60, do_sample=False)
        if isinstance(out, list) and len(out) > 0:
            summary = out[0].get("summary_text") or out[0].get("generated_text") or str(out[0])
            return summary.strip()
        return str(out)
    except Exception as e:
        # fallback stub
        return generate_summary_stub("Movie", keywords=None)
```

THANK YOU FOR LISTENING