(Group Patotoya) MindMelt App CS601P 5/27/2025

Members:
Pidlaoan, Cedric Vhon
Pabingwit, Nathan
Borromeo, John Michael
Selloria, John Fritz
Tirambulo, Morel

1. Project Overview

- **Brief description of the project**
  This project is a dedicated application designed to catalog, analyze, and share information about internet trends colloquially referred to as "brainrot." The term "brainrot" describes viral, often absurd or surreal online content that spreads rapidly across social media platforms, influencing digital culture in unexpected ways.

  The application serves as a centralized hub where users can
  Discover and contribute to a growing database of brainrot trends.
  Track the evolution of these trends over time.
  Engage with explanations, memes, and discussions surrounding them.
  Share their own findings or interpretations with a community of like-minded users.

- **Goals or purpose**
  Create an engaging platform dedicated to celebrating and sharing the most entertaining and absurd corners of internet culture
  Foster a community where users can collectively enjoy and contribute to the ever-evolving world of online humor
  Preserve fleeting digital phenomena that might otherwise disappear from the internet's collective memory
  Encourage creative expression through meme creation, trend participation, and humorous content sharing
  Develop an intuitive system that makes discovering new "brainrot" content effortless and enjoyable
  Promote positive interactions by maintaining a lighthearted, inclusive environment for all users
  Bridge generational gaps in humor by documenting how internet comedy evolves over time

Provide a space where stress relief and entertainment go hand-in-hand through shared laughter

Main features

Post Whatever

- Share your most brain-rotting pictures with a caption.
- Edit or delete your own mess anytime.
- Leave weird/funny comments on others' posts.

My Cringe, My Rules

- Check out your own profile and tweak it if you're feeling cute.
- Scroll through your collection of unfiltered posts.
- (Optional) Stalk—I mean—explore other users' masterpieces.

2. Installation / Setup Guide

System requirements

OS:

- Windows: 11 (24H2/23H2/22H2 Ent/Edu), 10 (22H2+), Server 2012–2025
- macOS: 13 Ventura, 14 Sonoma, 15 Sequoia
- Linux: Various distros (x64, x86, ARM64/32)

Hardware:

- CPU: ARM64 or x64 (quad-core+ recommended)
- RAM: 4 GB (16 GB recommended)
- Storage: 850 MB–210 GB (20–50 GB typical)
- Display: 1366×768 minimum

Dev Tools:

- Visual Studio: 2022 Preview 17.8.0+
- .NET CLI: Supported without IDE

How to clone or download the project

Option 1: Clone via GitHub (Recommended)

To clone the project directly into Visual Studio:

- Open Visual Studio 2022.
- Go to File → Clone Repository.
- In the Repository Location field, enter the following URL:
    - https://github.com/zed-koba/FinalProjectCSELE2.git
- Choose a local path where you want the project to be stored.
- Click **Clone.** Visual Studio will automatically fetch and load the project.

Option 2: Manual Download (ZIP)

Visit the repository page:

- **https://github.com/zed-koba/FinalProjectCSELE2**
- Click the green Code button, then choose Download ZIP.
- Extract the ZIP file to a preferred folder.
- Open the .sln file using Visual Studio 2022.

Option 3: Clone via Git Command (for advanced users)

- If you have Git installed on your machine, run the following command in your terminal or Git Bash:
    - git clone **https://github.com/zed-koba/FinalProjectCSELE2.git**
    - Note: If the repository is private, you will need to log in using your GitHub account credentials that have access to the repository.

How to install dependencies

Required Dependencies

- Mopups – for modal popup support
- CommunityToolkit.Maui – provides cross-platform UI helpers and utilities
- FFImageLoading.Maui – for advanced image caching and loading

Installation Steps

Option 1: Using NuGet Package Manager UI

- Open the solution in Visual Studio 2022.
- Right-click the project in Solution Explorer, then select Manage NuGet Packages.
- Go to the Browse tab and search for each of the following packages:
    - Mopups
    - CommunityToolkit.Maui
    - FFImageLoading.Maui
- Click Install for each package.
- Accept any license agreements when prompted.

Option 2: Using NuGet Package Manager Console

- You can also install the packages via the Package Manager Console (Tools > NuGet Package Manager > Package Manager Console):
    - Install-Package Mopups
    - Install-Package CommunityToolkit.Maui
    - Install-Package FFImageLoading.Maui
- Note: Make sure to clean and rebuild the solution after installing the dependencies to avoid any build issues.

How to run the project
Prerequisites

- Visual Studio 2022 with the following workloads:
- .NET Multi-platform App UI development (.NET MAUI)
- Android development (for emulator or device deployment)
- .NET SDK 8.0 or later
- A physical Android device (with USB debugging enabled) or an Android emulator configured via Android Device Manager
- Stable internet connection

Running the Application

Option 1: Run on Android Emulator

- Open the project in Visual Studio 2022.
- Select the target platform (e.g., Pixel_5_API_33 Emulator) from the debug toolbar.
- Click Run (▶ Start Debugging) or press F5.

- The app will be built and deployed automatically to the selected emulator.

Option 2: Run on Physical Android Device

- Connect your Android phone via USB.
- Ensure Developer Options and USB Debugging are enabled on the device.
- Your device should appear in the Visual Studio debug target list.
- Select your device and click Run or press F5.

3. Usage Instructions

How to Use the Application

Sign Up / Log In

- Create an account by filling out the required details.
- Log in using your registered credentials.

Profile

- Set up your profile with a username and avatar.
- Edit your profile anytime.
- View your own posts from your profile page.

Posting

- Share images with captions.
- Edit or delete your own posts.
- View and comment on posts from other users.

Walkthrough of features or UI

1. Login / Sign Up Screen

- Users land on a simple screen with options to Log In or Sign Up.
- New users can register by providing their username, email, and password.
- Existing users can log in to access their feed.

2. Home Feed

- After logging in, users are directed to the Home Feed.

- **Here, they can:**
    - Scroll through posts made by other users.
    - Tap on a post to view full image, read captions, and see comments.
    - Leave a comment under any post.

3. Create Post

- A visible "+" button or Post tab lets users create a new post.
- **Users can:**
    - Upload an image.
    - Add a caption.
    - Submit the post to appear on the feed.

4. Post Management

- On their own posts, users can:
- Edit the caption or image.
- Delete the post if they no longer want it shown.

5. Profile Screen

- Accessible via a profile button (top navigation bar).
- **Shows:**
    - User's avatar, username, and optional bio.
    - A list/grid of the user's own posts.
    - Option to edit profile (change avatar, user details).

- **Sample inputs/outputs if applicable**

4. Tech Stack

- **Languages, frameworks, libraries, tools used**
  C#, .Net Framework 8.0, XAML, .Net MAUI, Visual Studio 2022, CommunityToolKit, FFImageLoading, Mopups, CodeMaid(Extension/tool), JSON Formatter(Extension/tool)
- **Reasons for choosing them (optional)**

5. Project Structure

File/folder organization

Views

- Contains all the UI components (pages, user controls, or screens).
- Responsible for displaying the interface and receiving user input.
- Example: **MainPage.xaml, Registration.xaml, Homepage.xaml**

ViewModel

- Holds the logic and data-binding between the UI (View) and the backend data (Model).
- Processes user actions, updates data, and reflects changes to the view.
- Example: **AuthenticationViewModel.cs, PostViewModel.cs**

Model

- Defines the data structures used in the app.
- Contains classes that represent user data, posts, or other entities.
- Example: **AuthenticationModel.cs, PostModel.cs**

Features

- Stores helper functions or utility classes that enhance functionality.
- For example, the TimestampConverter.cs file help convert date/time formats (like converting Unix timestamps to human-readable format).
- These are reusable across the app for formatting or calculation purposes.

What each major file/module does

View

-These files define the user interface (UI) components.

- AddNewComment.xaml – UI for adding a new comment on a post.
- EditAvatar.xaml – Screen for changing or uploading a new profile avatar.
- EditPost.xaml – Interface to edit the content or image of an existing post.
- EditProfile.xaml – Screen for updating user information like name or bio.
- Homepage.xaml – Main feed showing all posts.
- LoadingScreen.xaml – Displayed while the app or data is loading.
- LoginViewLoading.xaml – Loading screen specific to the login process.
- OptionsPopup.xaml – Popup with options (e.g., log out, settings).
- Post.xaml – View to create and view individual posts.
- PostComment.xaml – View for adding and viewing comments on a post.
- ProfileOptionsPopup.xaml – Popup menu for profile-related actions.
- ProfilePage.xaml – Displays user's profile info and their posts.

- Registration.xaml – UI for user registration.

ViewModel

-These handle application logic and connect Views to Models.

- AuthenticationViewModel.cs – Logic for login and registration functionality.
- PostCommentViewModel.cs – Handles logic for commenting on posts.
- PostViewModel.cs – Manages post creation, editing, and retrieval.
- ProfileViewModel.cs – Manages user profile data and actions (e.g., edit profile, load posts).

Model

-These represent the data structures used in the app.

- AuthenticationModel.cs – Defines data for login and registration.
- CommentModel.cs – Represents a user comment.
- LikeModel.cs – Represents likes on a post (if implemented or for future use).
- PostModel.cs – Represents a post, including image, caption, and metadata.
- UserInteractionModel.cs – Tracks or defines interactions like likes, follows, etc.
- ViewAllCommentsModel.cs – Holds the structure for displaying multiple comments.
- ViewAllPostModel.cs – Holds the structure for displaying multiple posts.

Features

- Utility/helper functions that enhance app functionality.

- TimeStampConverter.cs – Converts Unix timestamps to readable date/time format and vice versa, used to display when posts or comments were made.

6. API Documentation (if applicable)

- **Endpoints, methods, parameters**

  MockAPI

Base URL : https://6821fa55b342dce8004c96f3.mockapi.io/

Endpoints

Endpoint	HTTP Method	Parameters	Description
`/UserDetails`	GET	None	Retrieve all users
`/UserDetails/{id}`	GET	`id` (path parameter)	Retrieve user by ID
`/UserDetails`	POST	JSON body (user data)	Create a new user
`/UserDetails /{id}`	PUT	`id` (path), JSON body	Update user by ID
`/UserDetails /{id}`	DELETE	`id` (path parameter)	Delete user by ID

Imgur API

Base URL : https://api.imgur.com/3/image

Upload Image Endpoint

HTTP Method	Parameters	Description
POST	`image` (base64 or binary)	Image file to upload
	`type` (string, e.g., "base64")	Specify type of image data
	`title` (optional)	Image title

- **Authentication requirements**

  MockAPI

  - No authentication or token required for this MockAPI.
  - Use endpoints directly with the base URL.

  Imgur API

  - Requires Client-ID in request header.
  - Header format:
    - Authorization: Client-ID YOUR_CLIENT_ID

- **Sample requests/responses**

  MockAPI

```
using System;

using System.Net.Http;

using System.Threading.Tasks;
```

```
class Program
{
   static async Task Main()
   {
      using HttpClient client = new HttpClient();
      var response = await
client.GetAsync("https://6821fa55b342dce8004c96f3.mockapi.io/users");
      var json = await response.Content.ReadAsStringAsync();
   }
}
```

Imgur API

```
var client = new HttpClient();
client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Client-ID",
"YOUR_CLIENT_ID");
var base64Image = Convert.ToBase64String(File.ReadAllBytes("path_to_image.jpg"));
var content = new MultipartFormDataContent();
content.Add(new StringContent(base64Image), "image");
content.Add(new StringContent("base64"), "type");
var response = await client.PostAsync("https://api.imgur.com/3/image", content);
var jsonResponse = await response.Content.ReadAsStringAsync();
```

7. Known Issues / Limitations

- **Current bugs or incomplete features**

  Slow Image Caching

  - Image loading may be delayed due to limited or inefficient caching mechanisms, especially when images are fetched from external sources like Imgur.

Slow Data Fetching

- Data retrieval can be slow under the following conditions:
    - Poor or unstable internet connection
    - Large volume of data being fetched simultaneously

Performance Issues with Large Datasets

- The application may experience lags or decreased responsiveness when handling a large amount of stored data, particularly if optimization techniques (such as pagination, lazy loading, or efficient data structures) are not implemented.

- **What doesn't work (yet)**
    - Forgot Password Feature (not completed nor started)

8. Future Improvements

Planned Enhancements

- Implement the ability to delete and edit comments
- Add upvote/downvote functionality for posts

Stretch Goals

- Introduce private messaging between users
- Enable friend or connection requests (add friends feature)
- Support video media uploads, if feasible

9. Credits / Contributors

- **Team members**
  Cedric Vhon Pidlaoan – Full Stack Developer
  John Michael Borromeo – Frontend Developer
  Morel Tirambulo – UI Designer
  John Fritz Selloria – Documentalist
  Nathan Pabingwit – Flower Boy

- **External resources or libraries used**

Libraries / Packages (via NuGet)

Name	Purpose	Author / Source
Mopups	Modal popups in .NET MAUI	Jonathan Peppers
CommunityToolkit.Maui	UI utilities and controls for .NET MAUI	.NET MAUI Community Toolkit Team
FFImageLoading.Maui	Fast image loading with caching support	Daniel Luberda

API and Services

Name	Purpose	Author / Provider
MockAPI	Simulated backend for testing	mockapi.io team
Imgur API	Image upload and hosting service	Imgur Inc.

Fonts and Icons

Name	Purpose	Author / Source
Poppins	Primary font used for UI	Indian Type Foundry, Google Fonts
Font Awesome	Icon set for UI elements	Fonticons, Inc. (fontawesome.com)