NEA Computer Science Proposals:

Webpage

Cloud based website where uses can upload their own music/recordings which are stored on a server. The webpage should include a modern and easy to use graphical user interface, geared towards 20–40-year-olds. Where applicable, YouTube embedded videos should be included, and a simple login system to save favourite tunes and music.

Should be done using API to communicate front end language loading the website with the back end, which will aggregate SQL functions to request data based on user requests.

If possible, encryption software should be done to protect accounts and login details. Also, if possible, a personalised section which will recommend certain music and recordings based on the user’s previous history done through a priority algorithm. (Time spent on certain types of music, how often a category of music is listened to).

Exercise App

Multiuser competitive exercise app that revolves around competing against others in gaining exercise points. Geared towards teenagers and young adults who are active or looking to become more active. It should be based around a daily leader board system where the person with the most exercise points that day gains a medal. There should also be a leader board for the people with the most medals overall. I will aim to program it using android studio in Java.

Should include an exercise timer with visual images built into the app or embedded YouTube videos for people to follow exercises, which will add more points to the user. Also, a viewable history of previous exercises which includes time spent on exercise and how many points were gained.

Should also have a simple graphical user interface with basic buttons, menu and scrolling functions.

The app may be able to use the location of a device to help track an exercise. (May require a hash algorithm and the Diffie Hellman protocol when exchanging data between a device and a server). If possible, a social networking system could be put in place to have a leader board with friends.

If possible, a personalised feed of exercises should be recommended on a page based off of previous exercises and age etc. API and parsing JSON for extracting and sending data about user scores to a server.

Client Server Model

API and parsing JSON for extracting and sending data about user scores to a server.

Aggregation of SQL functions.

Linked Lists to store details of exercises.

OOP to contain data about the user such as name, overall points, date of installation.

Diffie Hellman protocol to encrypt data about the location of a device (if possible)

Pygame Student themed Tycoon Game

Single player tycoon game based on the life of a student. It should involve a currency/point system where the student can use their points/currency to purchase new upgrades to increase their productivity in producing more points and currency. The game should include fundamental items such as a desk, gym, cinema, school, computer. These should then be upgradable to help productivity. This is geared towards teenagers and young adults who enjoy casual gaming.

Should include a graphical user interface and menu navigation. It should also include a time factor, where time spent away is considered being productive, but at a lower rate. (This may be manipulated by device times, so ideally if time element could be extracted via API using a json file)

If possible, it may be portable and distributable to different devices (windows) without the need for an IDLE.

If possible, a social networking feature should be available to add other users as friends and being able to view their current score and level.

API via JSON to communicate with a database on the top players with their associated scores.

2-d array manipulation to store data about given levels/upgrades of the game as well as graphs to represent social networks between users.

File manipulation to store details about specific upgrades.

Complex OOP to create the user objects with details about score, level. Other objects may be the upgrades available as well as the graphical images corresponding to the upgrade that should appear on the screen.