CEng 445 Project Topics

2019-2020 Fall

October 20, 2019-V1.0

Contents

1 Phases

The phases of the project will be:

1. Class library
2. Client–Server application
3. Web application with static content
4. Dynamic web application

In **class library** phase you should implement your basic classes and write a command line testing application demonstrating all features of your library.

In **client–server application** phase you should implement a server application creating instances of your class library and let clients connect and interact with them. You can use a simple line based or JSON based protocol that you will define for client–server communication. Also you need to implement a console based test application as the client connecting to your server. Note that this phase should respect concurrency of the shared instances. You need to use mutexes and other facilities to provide integrity.

In **web application** phase you need to write a wsgi or Django based web application to interact with a browser. This phase can be realized as a HTML form based wrapper for phase

2 or 1. The pages do not need to have fancy formatting and all operations can be on "server generates HTML, user fills in form,submits a button" loop.

In **dynamic web application** phase you need to use JavaScript, AJAX and web sockets to enhance the user experience. You can use HTML5 features like canvas, JQuery and other JavaScript widgets for better visualization (ask instructor for the libraries other than JQuery). All application will be loaded in a single HTML and rest is handled by JavaScript. You can either directly connect a version of phase 2 server or keep the web application layer in phase 3 as a web service (instead of HTML generating application). You need to use web sockets whenever appropriate

2 Sample Topics

2.1 A Borrowing and Storage System for Books, Movies and Albums

Project aims to help people keeping their books, DVDs and CDs in an organized manner. The type of items are not limited with those above but borrowing should make sense. The owner should be able to enter his/her belongings in the system with their locations (i.e. room/shelf), mark their borrowing status (borrowable, only borrowable to friends, not borrowable, borrowed).
Users of the system can search items of others and request borrowing, comment, watch, and rate. Owner can update status of the item when borrowed and returned. System should imeediately notify users for borrow requests, if a watched item becomes borrowable, and generate some basic reports.

2.2 A Rule Based Online Board Game

Project aims to create a generic board game engine that rules of the game are loaded as input along with the game configuration. The games to be played are in the class of children games played in turns with dice, most complex being Monopoly. Board will consist of cells either cyclic or linear. Each cell may be tagged by a movement rule (jump forward/back, wait for some number of turns etc), player option i (i.e. buy or skip, pay), modify credit, picking a card etc. Card sets can contain simple rules about credit or movement as well.

Once a game is loaded, multiple users can join and play in turns in a concurrent web based environment.

2.3 An Online Auction System

Project aims to develop a shared environment where people sell goods with an auction mechanism. Everyone has a virtual credit (not connected to a real payment mechanism of course) and bid on items that are announced on market by others. When seller and bidder agreed on

an item price, item is sold to the bidder.

For auctioning, at least 3 methods are to be implemented:

1. Highest bidder buys.
2. Price automatically decreases by time, first bidder buys.
3. Bidders immediately contribute to the accumulated sum, highest contribution buys (useful in charity auctions).

Seller should be able to setup some parameters like thresholds for automatic completion of auction. Once a bid started all online users should participate with real time notifications.

2.4 A Train Simulator Game for Kids

Project aims to implement a train simulator game similar to Lego Loco. Game has a grid arena and player places railroad pieces to create tracks for trains. Certain components generate trains during simulation, and trains start moving on tracks.

Each grid cell can contain only one component. Components consist of straight tracks, 90 degrees turns, switching junctions (track splitting in or merging from two ways), level crossings (with signals), bridged crossings and stations.

A simple collaborative railroad editor will be implemented for project, and users should be able to run simulation with moving trains. For simplicity, editor mode and simulation will be isolated, but multiple users should be able to edit the railroad and view the simulation concurrently. During simulation, junction switches can be dynamically altered.

2.5 A Chain Reaction Machine Editing and Simulation Tool

A chain reaction machine is a group of physical objects like domino blocks, marbles, balls, springs, with their actions are linked together so that a starting touch will lead a series of reactions and all pieces act one by one until the last piece. In this project you will implement a simple chain reaction machine editor in 2D and simulate it using Pymunk physics library. Last phase project output should have a web based visual collaborative editor with at least 8 moving component types and a Python/JavaScript interaction to simulate the machine.

2.6 A Visual Query Environment for CSV Files

CSV files are simple text files to store tabular information. This tabular information can be manipulated by a database by text utilities of Unix/Linux like `join, cut, sort, awk, sed`, and batch operations can be defined by using pipes. However this utilities are not easy to use and have limited power.

This project aims to implement operations of some of those text utilities (sort, join, cut, grep) in Python and letting user compose database queries by using a visual editor. Once a chain (actually like a directed acyclic graph) applied on CSV files, query result can be read on the sink nodes.

For example a CSV file has student id and exam grades, another one has student id and student names. When those tables are sorted and joined on student id a complete table can be constructed. Than a new column having a simple arithmetic on the grades will calculate overall grade in a new column and new table is sorted in descending order of this new column. This can be defined visually. Than user can upload these two CSV files and download the resulting table on web.