CEng 445 Project Topics

2018-2019 Fall

October 22, 2018 - 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 Question Bank for Multiple Choice Tests in LaTeX

This project aims to create a database for test exam questions typeset in LaTeX. LaTeX is a document typesetting language used especially in scientific documents and professional publishing. It is used to typeset exams as well. Multiple choice exams are composed of many questions and creating shuffled versions of the same exam is required to avoid cheating. Also questions asked in the past should be kept track when a new exam is composed. In this project, a question bank application aims to help instructors to maintain multiple choice exams.

Question bank application should:

- Provide CRUD (create, read, update and delete) operations on questions and answers written in LaTeX.
- Keep chapter, topic, difficulty, origin of questions, and external files (figures etc.) they refer to.
- Let user select questions based on topic and difficulty to compose a new exam.
- Create shuffled exams and their answer keys.
- Save final exams and record latest date of a question taking part in a composed exam.

There are command line tools that compiles LaTeX source into pdf and other formats. The achievements in phases:

1. A class library for all CRUD operations of question bank and question selection.
2. A client server application connecting a server and carrying operations in first phase through a TCP socket. The composed LaTeX exam will be send through TCP in text.
3. A web application having static forms and editing of operations. Final exam will be send as PDF to browser.
4. A dynamic web application with online editing of questions interactively. The target LaTeX view will be refreshed through web sockets as user edits the questions.

2.2 A Simple Logic Circuit Desing and Emulator

A logic design circuit can be represented as a directed graph with logics gates at vertex positions. The vertex in-degree and out-degree is predefined by the gate. Out-degree is typically 1. This project aims to build an editor for combinational logic circuits. User can add/remove logic gates and edges to the graph to compose a circuit. Then the circuit will be simulated and truth tables will be created as asked by the user. In addition to standart and, or, xor, (and negated versions), and not gate, input switch, output led and output 7 segment display should be supported.

The achievements in phases:

1. A Circuit class enabling editing of circuit elements and connections. Given set of values for input switches, a circuit object will give values of output leds.
2. A client server application for accessing editing operations remotely.
3. A web application to edit circuit elements non-interactively. Elements will be modified through dialog boxes. The circuit state can be drawn on user demand (you can use graphviz or a python library substitute to generate circuit layout)
4. An interactive web application editing circuit on a canvas and visualizing circuit state. Switch changes will change output leds interactively.

2.3 A Collaborative Editor for Docbook/Mallard XML documents

This project aims to implement an editor for simple structured documents in Docbook or Mallard formats. This formats are based on XML and used especially in user manuals. Generic XML editing is too complex and beyond scope is this course. However a well defined restricted set of tags as in basic Docbook/Mallard is easier. The editor will provide loading/saving facilities for documents, and editing of document structure. An XML document is basically a tree structured document with each node is defined by a *tag*. Editor will allow insertion/deletion/modification of any node based on XPath specified selectors. Editing should be done collaboratively meaning concurrent access to same document should be allowed. You can use a decent XML library like xml.etree.ElementTree or lxml.

The achievements in phases:

1. A Document class that implements load/save operations, selective editing of nodes in XML tree with XPath selectors, rendering/converting XML into HTML form for viewing.
2. A client–server application that new documents can be created and accessed concurrently.
3. A Web application that enables load/save/edit documents on an indented form document structure with HTML preview. Page is generated and refreshed on each form submit.
4. A dynamic web application with semi WYSIWYG like interaction, changes by collaborated users are immediately seen.

2.4 An Image Annotation Tool for Area Based Access Control

Sometimes, you do want to share or broadcast an image while keeping some areas hidden for some group of people. A typical example is a sattelite image with military areas. Details of those areas should not be shown to civilians. When the image is downloaded by a military officer full image should be shown, however another user should get the image with the area censored (in black for example). A similar scenario can be considered for news photography. Some areas of the picture can contain violence that you do not want kids to see.

This project aims to develop such an annotation tool that gets an image and let user to define possibly overlapping areas on it. Each area can be assigned to a set of access control descriptions which we call security labels. When this image stored along with its security labels is to be downloaded, —depending on the downloader—, the areas will be censored, applying rules in the security label. You can use OpenCV library that has shape drawing primitives over image files for blacking out censored areas.

The achievements in phases:

1. A class library that provide CRUD operations of annotated images. Images can have arbitrary number of security labels. Each label has an area description and list of users and groups that are allowed/denied access for that area. The area can be rectangular, in ellipse shape or a polyline.
2. A client–server application that carries same operation remotely on a TCP connection. Authentication is supported and only owner of image can edit it. The others can only download image which is subject to security label rules.
3. A Web application that enables editing of annotations through HTML forms, no interaction.
4. A dynamic web application with interactive editing of areas and security labels.

2.5 An Enhanced Log Watch Tool with Custom Filtering

Event logs of multiple systems can be sent to a remote log server for central management. Usually logs are text files without structure. Most of the logs are about usual activities and irrelevant for a system administrator. Filtering the logs for relevant events based on the content is essential in a central log management software.

In this project a small prototype of a customizable log management software will be developed. Logs comming from different hosts and facilities will be filtered and watched by user. User will be able to define custom rules for a log filter and incoming log source will be applied to these rules and matching log entries will be displayed in an online environment (immediately).

Each log filter object has its own set of rules to match a log entry based on:

- IP of the host generating the log
- Log severity (debug, info, notice, error, critical, etc)
- Log facility (mail, kernel, daemon, ...)

- Fields in log body (seperated by a predefine seperator)

A filter match can be defined on equality, inequality, substring or regular expression based. Filter rules can be combined with logic operators conjunction, disjunction and negation. In other words a filtering object ruleset will be a tree with rules at the leaves and logical operators in the nodes. User should be able to define a log resource (a file or a socket) and a ruleset in a filtering object. Then, as a new log matches the filter, it can be observed (read) by the user.

The achievements in phases:

1. A class library that has CRUD operations on filter objects, attaches to a log source, enables/watches selected filters on that.
2. A client–server application that carries same operation remotely on a TCP connection. Filters can be editted and attached to a log source and matching logs can be read on TCP connection.
3. A Web application that enables editing of filter objects. Matching logs will be displayed on user demand (press a refresh button)
4. A dynamic web application with interactive editing of filters and real time (server push) logs of matching filters.

3 Implementation Issues

You can and are encouredged to use basic libraries in your code. You do not need to write a image library, HTTP library or chart drawing library or similar for this project. However, you will not be allowed to use modules or source code from similar software. Please conduct instructor if you need to use a comprehensive library or software.

Implementation language is Python until the last phase. In last phase you can user JavaScript and JavaScript libraries on browser side. Server side should be python in last phase as well.