Project 1: Academic Reference Manager

The first project of the course is meant to put into practice designing and implementing a system, including API design and architecture.

Due date: 30 Sep 2019 (Canvas submission before 23:59)

Weight: 20% of final grade

Deliverables.

The solution to the project, i.e. code and design documentation. This should be submitted in Canvas (single ZIP file). Ensure that all files are included as part of the submission so the project can compile and run without the need of additional files. The TAs will compile and run the submission as part of evaluation and determining the grade.

For parts 2 and 3, you need to apply the ideas and concepts covered in the lectures. The code structure, code comments, choice of algorithms and data structures will be taken into account as part of the grade.

Note that this project is meant to be an individual project.

We expect students to know the University rules and codes regarding plagiarism. **Students are expected to submit their own solution.**

1. Prototype (40%): An Academic Reference Manager

You've started to like Usable Security as a research topic. You soon realize that the amount of literature out there is quite extensive and hard to keep track. You want to keep track of journals, papers and articles. In addition, you are subscribed to a few journals and magazines, you get printed publications.

Since you have a good collection of articles your friends have started to borrow the printed publications which include a variety of topics. To keep track of the borrowers you decide to create a simple prototype, the backend only, no UI at the moment.

The features of the prototype include:

1. Register the friends that borrow the publications, info:
    a. Name (first and last)
    b. Email
    c. Phone number
    d. Address
2. Register the publications
    a. Editor's name
    b. Title
    c. Journal
    d. Year
    e. Type: electronic or printed
3. Register when your friends borrow publications
    a. Borrower's identity
    b. Borrow Date
    c. Return Date
    d. Publication
4. You want to get reports
    a. For a particular date, output a **list of publications** that are on loan, and who borrowed them
    b. For a particular date, output a **list of friends** that borrow publications and which publications
    c. For a particular date, output a **list of friends** that have had borrowed a publication for more than a month

At this point you only want a prototype, thus command line interaction suffices, as mentioned above no UI implementation. Here are some assumptions you can work with:

1. The system does not need to store any information after the execution finishes.
2. Once a friend or publication has been registered you don't need to modify it
3. There are no duplicates of publications
4. Each friend only borrows the same publication at most once

Initialization.

Tue 10 Sep, at the latest, the instructor will provide two input sets.

Deliverable:

The project (code and deliverables from parts 2 and 3) as a ZIP file

2. Programmable Interface (30%)

Your above prototype was successful, and you want to continue enhancing the system. After writing the prototype you have a better understanding of the problem domain, thus you decide to properly design the system.

You realize that you need to define a programmable interface (API) for services. So, you define the following **services** and provide a RESTful API for each of them:

1. User Management
    a. Create, Read, Update, and Delete (CRUD)
2. Publication Management
    a. Create, Read, Update, and Delete (CRUD)
3. Lending Management
    a. Create, Read, Update, and Delete (CRUD)
4. Output report as specified above in part 1 removing the initial assumptions of the prototype

Adding a couple of requirements. You found that some of your friends have interesting insights into the papers and articles they are borrowing. So, you want them to rank the borrowed articles.

5. Article ranking
    a. Your friends can add a 1 to 5 star ranking to the articles
    b. They have full CRUD functionality to their own reviews
6. Top articles service
    a. Users can request a list of articles sorted by the rankins above, it should recommend articles that they haven't read

Please use Swagger to define the REST API and hand in a **YAML document** that runs and can be tested in Swagger or Postman.

Deliverable:

- YAML file, as part of the running program

3. Overall Design (30%)

You are happy with your work, you are considering making this an open source project. However, you realise more improvements can be supported. For example, supporting user types:

1. Admin Users (full access)
2. Authenticated Users (using OAuth for third party authentication) users can (CRUD personal information)
3. Anonymous Users (can only view your academic collection)

For this part the deliverables are:

- General description of the system (the reasoning behind your decisions: what, why, how, etc)
- Context, Container and Component Diagrams for the entire system
- The Class Diagrams for the Top articles service
- Schema for the database

It is important to note that all images and diagrams are required to have the proper descriptions and references in the text.

Deliverable:

- PDF document containing all the text, schema and diagrams