برنامه‌سازی پیشرفته

تمرین شماره‌ی هفت (تمرین نهایی)

مدرس: رامتین خسروی

طراحان: پویا مرادی، امیررضا دادفرنیا، محمد مهدوی‌دوست

این تمرین، آخرین تمرینی است که در این ترم انجام می‌دهید (تشویق حضار). قبل از تشریح صورت مسأله، توجه شما را به نکات زیر جلب می‌کنیم:

- این تمرین در قالب سه فاز طراحی شده‌است و هر یک از فازهای دوم و سوم، قابلیت‌هایی را به فاز قبلی می‌افزاید. در نتیجه لازم است طراحی شما به شکلی باشد که پس از تکمیل هر فاز، برای پیاده‌سازی صحیح فاز بعدی، نیاز به تغییرات زیادی نباشد.
- در انجام این تمرین لازم است همه ی مطالبی را که در طول این ترم در درس برنامه‌سازی پیشرفته آموخته‌اید، در نظر بگیرید. تأکید می‌کنیم که طراحی صحیح و رعایت اصول برنامه‌نویسی، به اندازه‌ی پیاده‌سازی کامل امکانات خواسته‌شده اهمیت دارد، و بنابراین به همان اندازه هم نمره دارد.
- بر خلاف تمرین‌های قبلی، صورت مسأله‌ی این تمرین، همه‌ی جزئیات را مشخص نکرده‌است. مثلا شکل ورودی و خروجی به طور دقیق بیان نشده و تنها به قابلیت‌های مورد نظر ما به صورت کلی اشاره شده‌است. دلیل این تغییر این است که انتخاب مواردی مثل شکل ورودی و خروجی و نحوه‌ی تعامل کاربر با نرم‌افزار، از اهمیت بالایی برخوردار است و لازم است که شما طراحی و پیاده‌سازی این گونه موارد را نیز به خوبی تمرین کنید. در نتیجه بخشی از نمره‌ی این تمرین به انتخاب صحیح این موارد اختصاص دارد.

اما صورت مسأله:

مجمع نویسندگان

در این تمرین قصد داریم یک سکوی[1] مناسب برای ایجاد، نگارش و نشر کتاب‌های دانشگاهی پیاده‌سازی کنیم. پیش از ورود به جزئیات هر فاز، با فرایندهای کلی این سامانه آشنا می‌شویم:

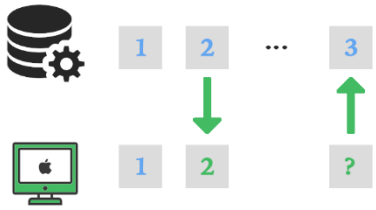
- جهت ایجاد یک کتاب جدید، دستور متناظر توسط یکی از کاربران سامانه وارد خواهدشد. حداقل اطلاعاتی که برای هر کتاب نگهداری می‌شود شامل عنوان و لیست نویسندگان آن است. کاربری که کتاب را ایجاد می‌کند، علاوه بر اینکه صاحب آن اثر است، می‌تواند در نقش یک نویسنده نیز ظاهر شود. توجه کنید که ویرایش عنوان و تغییر در لیست نویسندگان، در مراحل بعدی نیز امکان‌پذیر است، اما تنها توسط صاحب اثر. اگر این کتاب، اولین اثری باشد که کاربر فوق ایجاد می‌کند، دسترسی به آن نیازمند تایید مدیر مجمع است. در غیر این صورت، دسترسی به کتاب از همان ابتدا برای صاحب اثر و دیگر نویسندگان برقرار خواهد بود.
- هر کتاب از فصل‌های[2] متعددی تشکیل شده‌است که نویسندگان مشغول نگارش آنها هستند. نگارش هر فصل ممکن است به صورت انفرادی یا توسط چند نویسنده و به صورت هم‌زمان انجام شود. برای جلوگیری از تداخل، هر نویسنده

[1] Platform
[2] Chapter

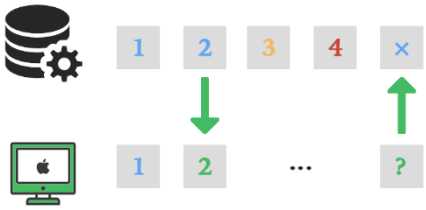
با وارد کردن یک دستور مشخص، آخرین نسخه‌ی آن فصل را از یک مخزن[3] مرکزی دریافت می‌کند، تغییرات مورد نظر خود را اعمال کرده و در نهایت نسخه‌ی(های) جدید را به مخزن ارسال می‌کند. در این مقطع با یکی از دو سناریو زیر روبرو می‌شویم:

۱. نسخه فعلی در مخزن، همان نسخه‌ای است که این نویسنده در ابتدا دریافت کرده بود، یعنی هیچ نویسنده دیگری، تغییرات خود را تا این لحظه به مخزن ارسال نکرده است: در این حالت، تغییرات این نویسنده در مخزن (به عنوان نسخه یا نسخه‌های جدید این فصل) اعمال می‌شود. تصویر شماره‌ی ۱، نمای کلی این سناریو را نمایش می‌دهد. کاربر سبز، آخرین نسخه (۲) از یک فصل دلخواه را دریافت می‌کند، تغییرات خود را روی آن اعمال می‌کند، محتوای تغییریافته را نسخه می‌زند (یک نسخه جدید ایجاد می‌کند) و این نسخه جدید را به مخزن ارسال می‌کند. از آنجاییکه در این فاصله، نسخه‌ی دیگری اضافه نشده‌است، تغییرات کاربر سبز در مخزن اعمال می‌شود و نسخه‌ی سوم فصل ایجاد می‌شود.

۲. در این فاصله یک یا چند نویسنده دیگر تغییراتی را در این فصل ایجاد کرده، نسخه‌های تغییر یافته را به مخزن ارسال کردند و تغییرات آنها در مخزن اعمال شده‌است: در این حالت، این نویسنده باید تمامی نسخه‌های جدید را از مخزن دریافت کند و به آخرین نسخه‌ی این فصل برسد. سپس تغییرات مورد نظر خود را مجددا اعمال کند و محتوای جدید را به مخزن ارسال کند. تصویر شماره‌ی ۲، نمای کلی این سناریو را نمایش می‌دهد. همانطور که مشاهده می‌کنید، در این فاصله، دو نسخه جدید توسط کاربر زرد و قرمز ایجاد شده و در مخزن اعمال شده‌اند. از این رو کاربر سبز باید ابتدا نسخه نهایی (۴) را دریافت کند، تغییرات خود را مجددا اعمال کند و محتوای جدید را ارسال کند.



تصویر ۱



تصویر ۲

- دقت کنید که ویرایش محتوای یک فصل به معنی ایجاد یک نسخه جدید از آن نیست. به عنوان نمونه این سناریو را در نظر بگیرید: نویسنده آخرین نسخه‌ی یک فصل را دریافت می‌کند، وارد محیط ویرایش می‌شود، بعضی از خطوط را تغییر می‌دهد و از محیط ویرایش خارج می‌شود. در این لحظه محتوای این فصل روی ماشین نویسنده تغییر کرده‌است، اما این تغییرات هنوز به یک نسخه جدید تبدیل نشده‌است. نویسنده مجددا وارد محیط ویرایش می‌شود، مجموعه دیگری از تغییرات را اعمال می‌کند و دوباره خارج می‌شود. این فرایند ورود، تغییر و خروج چند بار اتفاق می‌افتد تا زمانی که نویسنده از محتوای فعلی رضایت پیدا می‌کند و با وارد کردن یک دستور مشخص، محتوای فعلی را به عنوان نسخه جدید این فصل ثبت می‌کند. دقت کنید که این نسخه جدید روی ماشین نویسنده ایجاد شده‌است و مخزن مرکزی اطلاعی از آن ندارد. در این لحظه نویسنده می‌تواند با وارد کردن یک دستور دیگر،

[3] Repository

نسخه جدید را به مخزن مرکزی ارسال کند. با توجه به سناریو فوق ممکن است، یک نویسنده بیش از یک نسخه (برای یک فصل) روی ماشین خود جلو برود و همه آنها را هم‌زمان ارسال کند. در صورتی که نسخه‌های جدید روی ماشین نویسنده متعلق به بیش از یک فصل باشند، جدا کردن آنها هنگام ارسال امکان‌پذیر است، یعنی نویسنده می‌تواند فصل‌هایی را که دوست دارد، انتخاب کند و تنها تغییرات آنها به مخزن مرکزی ارسال خواهد شد.

- نسخه جدید یک فصل، یک کپی جدید از کل محتوای آن فصل نیست، بلکه تغییرات محتوای فعلی نسبت به محتوای قبلی را مشخص می‌کند.
- جهت ویرایش محتوای یک فصل، حداقل امکاناتی که در اختیار نویسندگان قرار می‌دهیم شامل اضافه کردن خطوط جدید، ویرایش خطوط قبلی و حذف خطوط مورد نظر است. توجه کنید که این فرایند درون برنامه شما اتفاق می‌افتد و چگونگی تعامل نویسنده با برنامه، توسط شما مشخص خواهد شد. جهت تسریع این فرایند، وارد کردن[4] محتوا از یک فایل txt. روی دیسک را نیز فراهم کنید.
- جهت ایجاد یک فصل جدید، صاحب کتاب یا هر یک از نویسندگان می‌توانند عنوان فصل را مشخص کرده و درخواست خود را وارد کنند. در ساده‌ترین حالت، فصل جدید، بدون هیچ محتوایی ایجاد شده و در انتهای کتاب قرار می‌گیرد. همچنین سازوکاری در نظر بگیرید تا بتوان محل قرارگیری فصل جدید را در کتاب مشخص کرد (مثلا بعد از فصل ۳).
- همان‌طور که در بخش‌های قبل اشاره شد، هر کتاب از تعدادی فصل تشکیل شده‌است. هر فصل نیز نسخه‌های مختلفی دارد که نمایان‌گر سلسله تغییرات نویسندگان روی محتوای آن فصل است. یکی از قابلیت‌های اصلی این سامانه، بازگشت به نسخه‌های قبلی یک فصل است. یک نویسنده می‌تواند روی ماشین خود، به هر یک از نسخه‌های قبلی فصل باز گردد. در نظر داشته باشید که با بازگشت به یک نسخه قدیمی، تمام نسخه‌های بعد از آن از بین خواهند رفت. فرایند بازگشت، در نسخه‌های موجود در مخزن نیز امکان‌پذیر است، اما تنها صاحب کتاب است که می‌تواند یک فصل را به نسخه‌های قبلی آن بازگرداند. دقت کنید که فرایند فوق، ابتدا روی ماشین صاحب اثر انجام شده و پس از تکمیل، تغییرات در مخزن مرکزی نیز اعمال می‌شوند.
- نویسندگان می‌توانند لیست فصل‌های یک کتاب و محتوای فعلی هر یک را مشاهده کنند. نمایش محتوای یک فصل، محدود به نسخه فعلی آن نیست و در صورت نیاز، باید بتوانیم بدون نیاز به بازگشت به نسخه‌های قبلی، محتوای هر یک از آنها را نیز مشاهده کنیم. مشاهده تغییرات یک نسخه نسبت به نسخه‌ی قبل نیز امکان‌پذیر است.
- پس از تکمیل نگارش کتاب، صاحب کتاب می‌تواند آن را تایید و سپس منتشر کند. در نظر داشته باشید که پس از انتشار کتاب، هیچگونه تغییری در محتوا یا اطلاعات آن، توسط نویسندگان یا صاحب اثر، امکان‌پذیر نخواهد بود. همچنین همه‌ی کاربران سامانه می‌توانند اطلاعات این کتاب را مشاهده و محتوای آن را به صورت یک فایل HTML[5] دریافت کنند.
- جهت استفاده از سامانه به یک شناسه کاربری یکتا و گذرواژه متناظر نیاز داریم. این مقادیر توسط کاربر هنگام ثبت‌نام مشخص خواهند شد. در صورت نیاز می‌توانید اطلاعات دیگری را نیز از کاربران دریافت کنید اما دقت کنید

[4] import
[5] http://www.w3schools.com/html/html_intro.asp

که فرایند ثبت‌نام طاقت‌فرسا نباشد و کاربران بتواند در صورت کمبود وقت، در مقاطع بعدی اطلاعات غیرضرروی خود را تکمیل کنند. مدیر مجمع نیز به صورت پیش‌فرض توسط سامانه ایجاد خواهد شد (لطفا از شناسه کاربری و گذرواژه‌های خیلی طولانی یا عجیب استفاده نکنید تا هنگام تست برنامه شما، دستیاران آموزشی دشواری زیادی نداشته باشند).

اگر دقت کرده باشید، زمانی که در مورد نسخه‌های مختلف فصل‌ها و کتاب صحبت می‌کردیم، به دو مفهوم مستقل اشاره کردیم: ماشین نویسنده و مخزن مرکزی. مساله‌ای که با آن روبرو هستیم این است که کاربران محتلف دوست دارند به صورت همزمان از سامانه ما استفاده کنند. اما تمام برنامه‌هایی که تا به این لحظه پیاده‌سازی کرده‌اید، تک کاربر بوده‌اند، یعنی یک کاربر خاص وارد سامانه می‌شد، عملیات مورد نظر خود را انجام می‌داد و تنها پس از خروج آن کاربر از سامانه، یک کاربر جدید می‌توانست وارد شود. در این تمرین قصد داریم با استفاده از معماری کلاینت - سرور[6]، بخش قابل توجهی از پردازش و نگهداری از داده‌های مشترک را به یک سرور مرکزی واگذار کنیم که به صورت همزمان به درخواست هر یک از کلاینت‌ها (کاربران) رسیدگی خواهد کرد.

واسط کاربری گرافیکی[7]، از دیگر قابلیت‌هایی است که در انتهای این تمرین برای کلاینت‌های خود پیاده‌سازی خواهید کرد. مجددا به برنامه‌های شما تا این لحظه اشاره می‌کنیم: تعامل میان کاربر و تک‌تک این برنامه‌ها با استفاده از خط فرمان (ترمینال) انجام می‌شد. یک گزینه جایگزین، واسط کاربری گرافیکی است، به این معنی که اطلاعات در یک محیط گرافیکی به کاربران نمایش داده خواهد شد و تعامل آنها با سامانه نیز با فشار دادن دکمه‌های مختلف و وارد کردن داده‌های مورد نظر در همان محیط انجام خواهد شد.

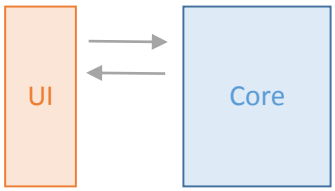
[6] Client-Server
[7] Graphical User Interface (GUI)

در ادامه قابلیت‌های مورد نظر برای هر فاز را به صورت جداگانه مشخص خواهیم کرد:

فاز اول، هسته‌ی برنامه

موعد تحویل: جمعه ۳۱/اردیبهشت/۱۳۹۵

در فاز اول این تمرین، هسته برنامه‌های سرور و کلاینت را پیاده‌سازی خواهیم کرد. هسته، موجودی‌است که منطق برنامه شما درون آن پیاده‌سازی می‌شود. دسترسی به هسته توسط یک اینترفیس انجام خواهد شد، یعنی مجموعه‌ای از توابع که امضای آنها مشخص است، اما چگونگی پیاده‌سازی آنها (از نگاه مصرف‌کننده) اهمیت خاصی ندارد. شکل زیر تمایز میان واسط کاربری و هسته برنامه را نمایش می‌دهد:



مستقل از اینکه واسط کاربری شما، خط فرمان است یا گرافیکی، دستورات کاربر از شبکه دریافت می‌شوند یا ورودی استاندارد، هسته شما نباید دستخوش تغییرات شود. در کامل‌ترین نسخه‌ی برنامه، برای هر یک از کاربران سامانه، به یک هسته کلاینت نیاز داریم در حالیکه تنها یک هسته سرور که به صورت مشترک بین همه کاربران استفاده خواهد شد و پاسخگوی نیازهای برنامه ما خواهد بود.

برنامه شما در این فاز تک کاربر خواهد بود، یعنی در هر لحظه فقط یک کاربر می‌تواند از برنامه شما استفاده کند. با اجرای برنامه، یک نمونه[8] سرور و یک نمونه کلاینت ایجاد می‌شود و رسیدگی به دستورات کاربر (که از خط فرمان دریافت می‌شوند) به کلاینت واگذار می‌شود. به عنوان نمونه، اولین پیامی که به کاربر نمایش می‌دهید، می‌تواند این پرسش باشد که قصد ورود دارد یا می‌خواهد به عنوان یک نویسنده جدید ثبت‌نام کند. با توجه به طراحی شما، ممکن است کلاینت برای رسیدگی به درخواست‌ها، با سرور تعامل داشته باشد. به عنوان نمونه نام کاربری و کلمه عبور وارد شده توسط کاربر را در اختیار سرور قرار دهد و با توجه به پاسخ سرور، پیام مناسب را به کاربر نمایش دهد. دقت کنید که با خروج یک کاربر از سامانه، تمام تغییرات او که در سرور ثبت نشده‌اند، از دست خواهد رفت.

در این فاز تمام قابلیت‌های ذکر شده در بخش مقدمه را پیاده‌سازی خواهید کرد.

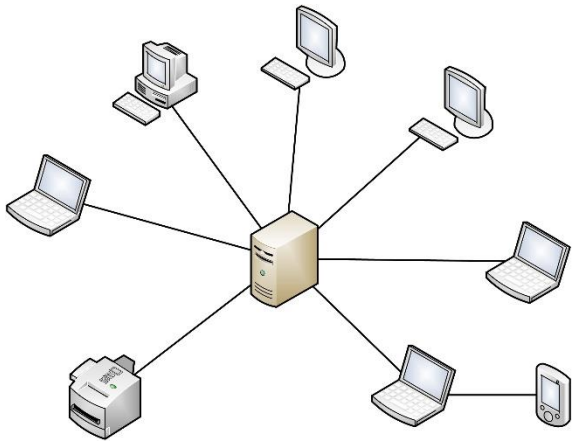
[8] instance

فاز دوم، شبکه **موعد تحویل: سه‌شنبه ۱۱/خرداد/۱۳۹۵**

قبل از بررسی جزئیات این فاز، توضیحی مختصر در مورد معماری کلاینت - سرور:

این معماری دسترسی همزمان کاربران به طیف گسترده‌ای از منابع شامل تصویر، ویدئو، فایل‌های متنی و ... را از طریق یک کامپیوتر مرکزی به نام سرور فراهم می‌کند. همان‌طور که از اسم سرور مشخص است قرار است سرویس‌های را در اختیار کاربران (کلاینت‌ها) قرار دهد و یا اصطلاحا کلاینت‌ها را serve کند. سرور مورد نظر ممکن است در یک کشور دیگر، روی یک کامپیوتر دیگر در شبکه محلی شما و یا حتی روی کامپیوتر شخصی خودتان در حال اجرا باشد.

احتمالاً تجربه‌ی بازی‌های آنلاین را دارید، برای مثال فرض کنید می‌خواهیم بازی فیفا را به صورت آنلاین بازی کنیم، در این حالت کامپیوتر یا کنسول بازی شما به یک سرور مرکزی متصل می‌شود، افرادی که قرار است با آن‌ها بازی کنید نیز از طریق شبکه‌ی اینترنت به همان سرور متصل می‌شوند، بقیه‌ی کارها و رد و بدل کردن اطلاعات بین بازیکن‌ها توسط سرور مرکزی انجام می‌شود. در این حالت شبکه‌ی ارتباطی ما اینترنت است.



در این فاز می‌خواهیم با داشتن یک سرور مرکزی امکان دسترسی همزمان بیش از یک کاربر را به برنامه فراهم کنیم. دقت کنید که در این برنامه، شبکه‌ی ارتباطی ما اینترنت نخواهدبود و تمام ارتباطات بر روی کامپیوتر شما (local) انجام می‌شود.

با استفاده از کتاب‌خانه‌ای که در اختیارتان قرار داده‌ایم، کلاینت و سرور در دو برنامه‌ی جدا از هم اجرا شوند. به این‌صورت که سرور همواره در حال اجراست و کلاینت‌ها (که همان ماشین شخصی هر نویسنده یا صاحب‌اثر است) در موقع نیاز با داشتن آدرس سرور به آن متصل می‌شوند، و اطلاعات لازم را از سرور دریافت و یا به آن ارسال می‌کنند.

کاربران فقط با کلاینت در ارتباط هستند، و تغییرات خود را به صورت محلی در ماشین شخصی خود (برنامه‌ی کلاینت) ایجاد می‌کنند. (تا زمانی که کاربر تصمیم به اعمال تغییرات خود بر روی مخزن نگرفته‌است، تمامی این تغییرات به صورت محلی ذخیره می‌شود و به سرور فرستاده نمی‌شود)

فاز سوم، رابط کاربری گرافیکی

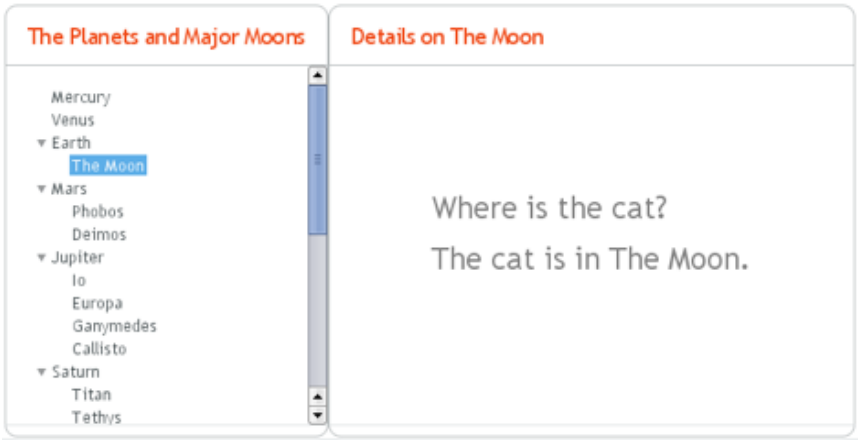
موعد تحویل: سه‌شنبه ۱۱/خرداد/۱۳۹۵

تک‌تک تمرین‌هایی که در طول این ترم انجام داده‌اید را در نظر بگیرید. غیر از تمرین سوم که با چارچوب[9] SDL آشنا شدید و رابط کاربری شما گرافیکی تلقی می‌شد، رابط‌های کاربری استفاده‌شده، همگی خط فرمان بودند. رابط کاربری خط فرمان برای هر نوع مخاطبی مناسب نیست، پس در انتخاب یک رابط کاربری، علاوه بر اصول برنامه‌سازی، مشتری خود را نیز ملاک قرار می‌دهیم. یک نمونه از چارچوب‌هایی که در پروژه‌های معتبر استفاده می‌شود، **Qt**[10] می‌باشد. مطمئنا با محصولاتی همچون VLC player، Telegram، Spotify آشنا هستید. هر یک از این نرم‌افزارها به نوعی از این چارچوب استفاده می‌کنند.

فاز سوم تمرین شما شامل طراحی و پیاده‌سازی یک واسط گرافیکی تحت چارچوب Qt برای کلاینت‌های این سامانه است. هدف از این فاز، بیشتر از اینکه یادگیری چارچوب Qt باشد، سنجیدن توانایی شما، در یادگیری و استفاده از یک کتاب‌خانه‌ی جدید است. چرا که این مهارت، یکی از مهم‌ترین قابلیت‌های یک فرد متخصص در حوزه کامپیوتر می‌باشد. بهترین راه برای یادگیری این چارچوب، جستجو در وب، مطالعه‌ی مراجع و یا بررسی ویدئوهای آموزشی می‌باشد. جستجوی عبارت‌های کلیدی مانند Qt menu tutorial یا Qt form tutorial نیز می‌تواند راهگشا باشد.

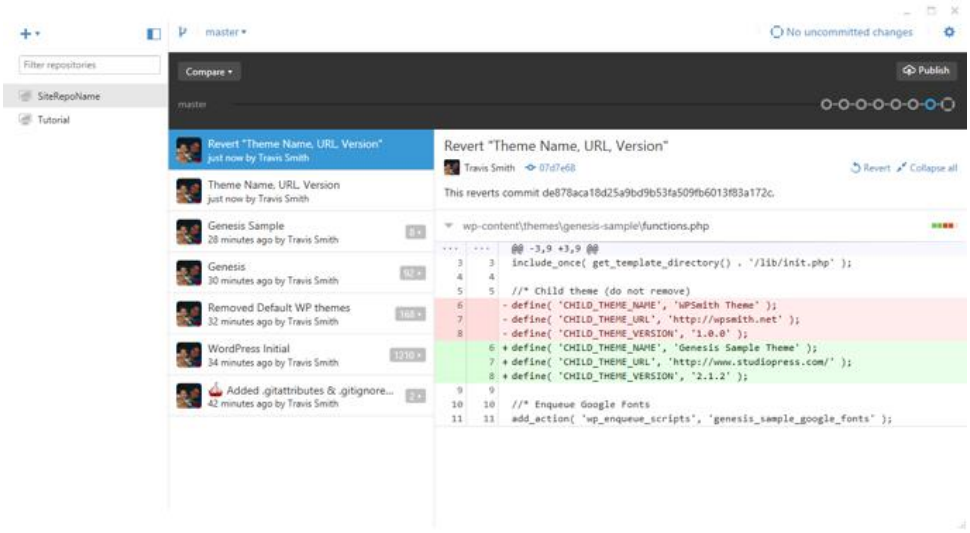
مهم‌ترین نکته در این فاز، مدیریت مناسب زمان می‌باشد پس بهتر است هر چه سریع‌تر با مفاهیم اولیه‌ی Qt آشنا شوید. همچنین در نظر داشته باشید که ما دستیاران آموزشی نیز بخش قابل توجهی از نکات مربوط به Qt را فراموش کرده‌ایم و نمی‌توانیم کمک‌تان کنیم! ملاک سنجش، کاربرپسند بودن محصول نهایی از دید یک مشتری است پس هر گونه کوتاهی در طراحی رابط کاربری مناسب، منجر به کسر نمره خواهد شد. حداقل امکانات مورد نیاز به شرح زیر است:

- در رابط کاربری گرافیکی شما، باید تمام امکاناتی که در فاز اول، پیاده‌سازی کردید، قابل دسترسی باشد. بر این مساله تاکید می‌کنیم که طراحی شما در فاز اول باید به گونه‌ای باشد که تغییر رابط کاربری، تاثیری در نحوه‌ی پیاده‌سازی منطق برنامه‌ی شما نداشته باشد. رابط گرافیکی، صرفا واسط بین هسته‌ی برنامه و کاربر است و از خود عملکردی (در ارتباط با منطق برنامه) ندارد.
- نمایش لیست کتاب‌های هر کاربر و فصل‌های آن به صورت درختی انجام شود:



[9] Framework
[10] https://en.wikipedia.org/wiki/Qt_(software)

- نمایش تغییرات یک نسخه نسبت به نسخه‌ی قبل به صورت ساختاریافته. به طور مثال می‌توانید خطوط جدید را با رنگ سبز، خطوط حذف‌شده را با رنگ قرمز، خطوطی که تغییر کردند را با رنگ آبی و خطوطی که ثابت ماندند را با رنگ مشکی نمایش دهید. این فرمت رنگ‌آمیزی فقط یک مثال بود و هر گونه نمایش مناسب، قابل قبول است.



نحوه‌ی تحویل

فایل‌های cpp. و h. خود را همراه با Makefile به برنامه‌ی خود را در یک پوشه‌ای به نام A7-SID-PHASE قرار دهید، آن را با فرمت zip آرشیو کنید و در نهایت فایلی با نام A7-SID-PHASE.zip را در سایت درس آپلود کنید. (SID پنج رقم آخر شماره‌ی دانشجویی شما و PHASE یکی از مقادیر ۱ تا ۳ خواهد بود)

دقت کنید

- در صورت وجود ابهام در ابتدا متن پروژه را دقیق مطالعه کنید و سپس اگر ابهام برطرف نشد در فروم درس سوالات خود را مطرح نمایید.
- برنامه‌ی شما باید در سیستم‌عامل لینوکس نوشته و با کامپایلر استاندارد g++ کامپایل شود.
- به فرمت و نام فایلهای خود دقت کنید .در صورتی که هر یک از موارد گفته شده رعایت نشود، نمره‌ی صفر برای شما در نظر گرفته می‌شود.
- در صورت کشف تقلب در کل و یا قسمتی از تمرین، برای هر دو طرف نمرهی ۱۰۰- منظور خواهد شد.