



دانشگاه آزاد اسلامی

واحد تهران شمال

دانشکده برق و کامپیوتر

پروژه پایانی کارشناسی کامپیوتر گرایش نرم افزار

عنوان

زمانبندی کلاس ها دانشکده با استفاده از الگوریتم ژنتیک

استاد راهنما:

دکتر سروش مبشری

تهیه و تنظیم:

سید علی مجتبوی - امیر حسین کفاشی - امیر رضا فصیحی راد

خرداد ۱۴۰۱



سپاسگزاری

پروردگارا مرا یاری کن تا دانش اندکم نه نردبانی باشد برای فزونی و تکبر و غرور، نه حلقه‌هایی برای اسارت و نه دستمایه‌ایی برای تجارت، بلکه گامی باشد برای تجلیل از تو و تعالی ساختن زندگی خود و دیگران. قبل از هر چیز، خداوند بزرگ را به خاطر لطفی که همواره شامل حال من نموده شاکرم. سپس، از زحمات استاد محترم راهنما، جناب آقای دکتر سروش مبشری که نه تنها به عنوان استاد بلکه همچون همکاری در تمام مراحل انجام این تحقیق از رهنمودها و کمک های بیدریغ ایشان بهره‌مند شده ایم، به ویژه به خاطر ساعتهای طولانی که به بحث و تبادل نظر در مورد موضوع تحقیق بنده اختصاص داده اند که همواره برای ما الهامبخش ایده و دیدگاهی تازه نسبت به موضوع بوده است، تشکر و قدردانی میکنیم.

فهرست مطالب

عنوان

صفحه

| | | |
|-----------------------------------|-------|------------------------------|
| چکیده | | Error! Bookmark not defined. |
| مقدمه | | ۹ |
| پیکره | | ۱۱ |
| فصل اول: چارچوب مفهومی نرم افزار | | ۱۲ |
| فصل دوم: اجزای تشکیل دهنده سیستم: | | ۱۴ |
| الگوریتم | | ۱۴ |
| رابط کاربری | | ۱۵ |
| دیتابیس | | ۱۵ |
| مدیریت سناریو | | ۱۶ |
| فصل سوم: ساختار پروژه | | ۱۷ |
| دایرکتوری Components | | ۱۷ |
| دایرکتوری Containers | | ۱۷ |
| دایرکتوری py_ui | | ۱۷ |
| فایل gas.db | | ۱۸ |
| فایل requirements.txt | | ۱۸ |
| فایل settings.json | | ۱۸ |
| فایل timeslots.json | | ۱۸ |
| فصل چهارم : ساختار ساخت زمانبندی | | ۱۹ |
| کتابخانه های مورد نیاز الگوریتم | | ۱۹ |

| | |
|----|--------------------------------|
| ۲۰ | کلاس کروموزم در الگوریتم ژنتیک |
| ۲۶ | GeneticAlgorithm کلاس |
| ۳۱ | متد selectRoom |
| ۳۱ | متد selectInstructor |
| ۳۲ | متد selectTimeDetails |
| ۳۳ | تابع ارزیابی |
| ۳۶ | تنظیم جمعیت |
| ۳۷ | تنظیم نرخ جهش |
| ۳۸ | انتخاب جمعیت هدف |
| ۴۰ | جفت گیری و تولید نسل جدید |
| ۴۱ | جهش |
| ۴۱ | دامنه و محدودیت |
| ۴۲ | خروجی نرم افزار |
| ۴۲ | تنظیمات الگوریتم ژنتیک |
| ۴۴ | فصل پنج : رابط گرافیکی |
| ۴۴ | ساختار کلی: |
| ۴۴ | پنجره اصلی: |
| ۴۴ | نمایش داده ها: |
| ۴۵ | ستون عملیات |
| ۴۵ | ستون در دسترس |
| ۴۵ | جدول زمانی |
| ۴۶ | اساتید |
| ۴۷ | کلاس ها |

| | |
|------------------------|----|
| درس ها | ۴۸ |
| گروه ها | ۵۰ |
| مدیریت سناریو | ۵۲ |
| ایجاد | ۵۲ |
| ساختار کد | ۵۳ |
| ابزار های مورد استفاده | ۵۳ |
| طراحی GUI | ۵۳ |
| اجزای GUI | ۵۴ |
| Py_ui | ۵۴ |
| Containers | ۵۴ |
| Instructor | ۵۵ |
| Subject | ۶۲ |
| Room | ۶۶ |
| Section | ۶۶ |

مقدمه

امروزه یکی از سخت ترین مشکلاتی است که دانشگاه ها و کالج ها با آن مواجه هستند، مسئله زمان بندی کلاس های دانشگاه (UCSP) است، که یک مسئله بهینه سازی بسیار محدود ترکیبی، در دنیای واقعی است. حل UCSP به معنای ایجاد یک برنامه بهینه با اختصاص دوره ها به اتاق های خاص، مدرسان، دانش آموزان و بازه های زمانی خاص با در نظر گرفتن محدودیت های داده شده است . UCSP یک مسئله زمان بندی کلاسیک است و NP-کامل در نظر گرفته می شود . مسئله های زمان بندی باید دو نوع محدودیت را برآورده کند، یعنی محدودیت های سخت و نرم که در آن محدودیت های سخت شرایطی هستند که برای یک جدول زمانی کاری حتما باید رعایت شوند، در حالی که محدودیت های نرم شرایطی هستند که ممکن است نقض شوند اما بر کیفیت راه حل تأثیر می گذارند.

برنامه ریزی کلاس ها در یک موسسه یکی از کارهایی است که می تواند تحت دسته بندی "مدیریت عملیات" طبقه بندی می شود. اهداف مدیریت عملیات برای به حداکثر رساندن کارایی در یک زمینه مشخص استفاده خواهد شد. برنامه ریزی کلاس ها اغلب

توسط مدیر گروه (نیروی انسانی) انجام می شود. تا کارایی لازم را داشته باشد. ولی این به معنی بی نقص بودن برنامه نیست. استفاده از کامپیوتر برای یافتن بهترین راه حل برای این مسئله ها با استفاده از روش های مرسوم بسیار ناکارآمد است. بنابراین کامپیوتر های برای مدتی برای حل این مسئله کنار گذاشته شده بودند با این حال، افزایش قدرت محاسبه کامپیوتر ها و استفاده از الگوریتم های جدید، راهی را برای حل مسئله های زمانبندی باز کرده است.

مدیریت عملیات اغلب به عنوان ستون فقرات و شاکله بسیاری شرکت ها از در نظر گرفته می شود. به دست آوردن کارآمدترین عملکرد به معنای سود بیشتر است. در یک سناریوی مؤسسه آموزشی، به دست آوردن کارآمدترین برنامه زمانبندی، نه تنها به کاهش هزینه ها کمک می کند، بلکه عمدتاً به دانشجوی کمک می کند. در بیشتر مواقع برنامه مناسب و کارآمد به دانشجوی ها و مربیان کمک می کند تا کنترل بهتری بر زمان خود داشته باشند. ایجاد برنامه زمانی توسط انسان ها بهتر از روش های تکنیک های محاسباتی معمول انجام می شود زیرا انسان ها دارای مغز قدرتمندی برای ارزیابی محدودیت ها و ترکیب ها هستند.

با این حال، مسئله های زمانبندی به عنوان **non-deterministic polynomial time (NP)** شناخته می شود. که بهترین راه حل برای آنها هنوز پیدا نشده است و برای یافتن بهترین آن، باید هر ترکیب ممکن را اجرا کرد. انسان ها نمی توانند هر ترکیب ممکن را محاسبه کنند و بنابراین با پر کردن و اطمینان از رعایت محدودیت ها مشکل را حل می کنند. این راه حل ها به اسم "به اندازه کافی خوب" شناخته شده است. این فرآیند مستعد خطا، ناکارآمدی و نقض محدودیت ها است، به ویژه زمانی که در یک سناریوی بسیار فشرده کار می کنید. تکرار روش انسانی برای حل مسئله (الگوریتم Greedy) در رایانه به دلیل هزینه محاسباتی آن بسیار ناکارآمد است.

کامپیوترها به ندرت برای حل این مشکل استفاده می شوند، زیرا غیر قابل اعتماد و ناکارآمد هستند. با این حال، این به زمانی برمی گردد که کامپیوترها کند بودند و برای حل مشکلات محاسباتی در نظر گرفته نمی شدند. با انفجار کلان داده ها و علوم داده و^۳ افزایش تصاعدی قدرت محاسباتی، استفاده از رایانه ها برای حل مسائل مجموعه های جدیدی که قبلاً فقط توسط انسان قابل حل بود، آسان شد. زمان بندی یکی از مسائلی است که هم اکنون توسط رایانه ها با راه حل های قابل قبول یا حتی بهتر قابل حل است. هوش مصنوعی امروزه به دلیل رشد صنعت محاسبات رایج تر شده است. چندین راه حل مانند یادگیری ماشینی، یادگیری عمیق و الگوریتم های ژنتیک تحت هوش مصنوعی قرار می گیرند. همه این فرآیندها مزایا و معایب خاص خود را دارند. الگوریتم ژنتیک برای مسئله زمانبندی مناسب است زیرا در طول زمان بر اساس قوانین و معیارها راه حلی جدیدی ایجاد می کند.

در دانشگاه آزاد اسلامی، مدیر گروه مسئول ایجاد زمانبندی است. زمان بندی به صورت نیمه دستی با کمک کامپیوتر انجام می شود. این فرآیند هنوز نمی تواند راه حلی ارائه دهد که مطمئن باشد از خطاها، نتایج بد ظاهری و نقض محدودیت ها عاری باشد. با شرایط

^۱ good enough

^۲ Big Data

^۳ Data Science

فعلی استفاده از کامپیوتر برای کمک به حل مشکل زمان بندی می تواند یکی از گزینه ها باشد. هدف این پروژه ایجاد یک راه حل نرم افزاری کاندید با استفاده از الگوریتم های ژنتیک برای تولید زمان بندی است.

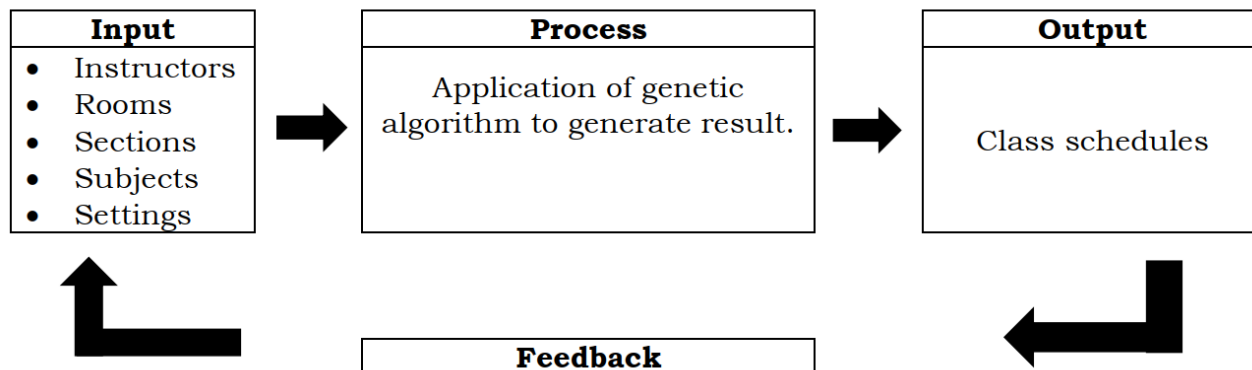
پیکره

برنامه ریزی دانشگاه آزاد اسلامی به صورت نیمه دستی با کمک کامپیوتر انجام می شود که می تواند خطاها، نقض محدودیت ها و ناکارآمدی ایجاد کند مسئله را می توان به تقسیم کرد؛

- زمان بندی هنوز به صورت نیمه دستی با اپراتور انجام می شود که این فرآیند به بسیار شبیه الگوریتم حریصانه است که می تواند نتایجی با خطا ایجاد کند.

فصل اول: چارچوب مفهومی نرم افزار

این برنامه را می توان به راحتی به عنوان یک نرم افزار محاسباتی زمان بندی توصیف کرد که در آن مجموعه داده های اولیه را وارد می کنید و یک نتیجه ساختاریافته به دست می آورید. نمودار جریان داده نرم افزار به اینصورت خواهد بود:



الگوریتم ژنتیک یک روش محاسباتی است که انتخاب طبیعی را انجام می دهد که فرآیندی در تکامل بیولوژیکی است. این فرآیند تکراری است و با استفاده از نمودار جریان به بهترین شکل مفهوم سازی می شود. در الگوریتم ژنتیک، به یک محلول به جای انفرادی، کروموزوم گفته می شود. شکل زیر فرآیند اصلی الگوریتم ژنتیک را نشان می دهد اما در عمل به سیستم، مرحله دیگری پس از ارزیابی اضافه می شود که تنظیم محیط^۴ است.

تولید جمعیت باید به طور تصادفی با ترکیبی از رویکرد حریصانه انجام شود که در آن یک نقطه تصادفی انتخاب می شود و به عنوان نقطه شروع عمل می کند و سپس شروع به پر کردن جدول با هر موجودی مناسب می کند. در طول تولید جمعیت، همه محدودیت های سخت باید برآورده شوند. محدودیت های متوسط چندین بار برای پیگیری انجام می شوند در حالی که محدودیت های نرم کاملاً نادیده گرفته می شوند.

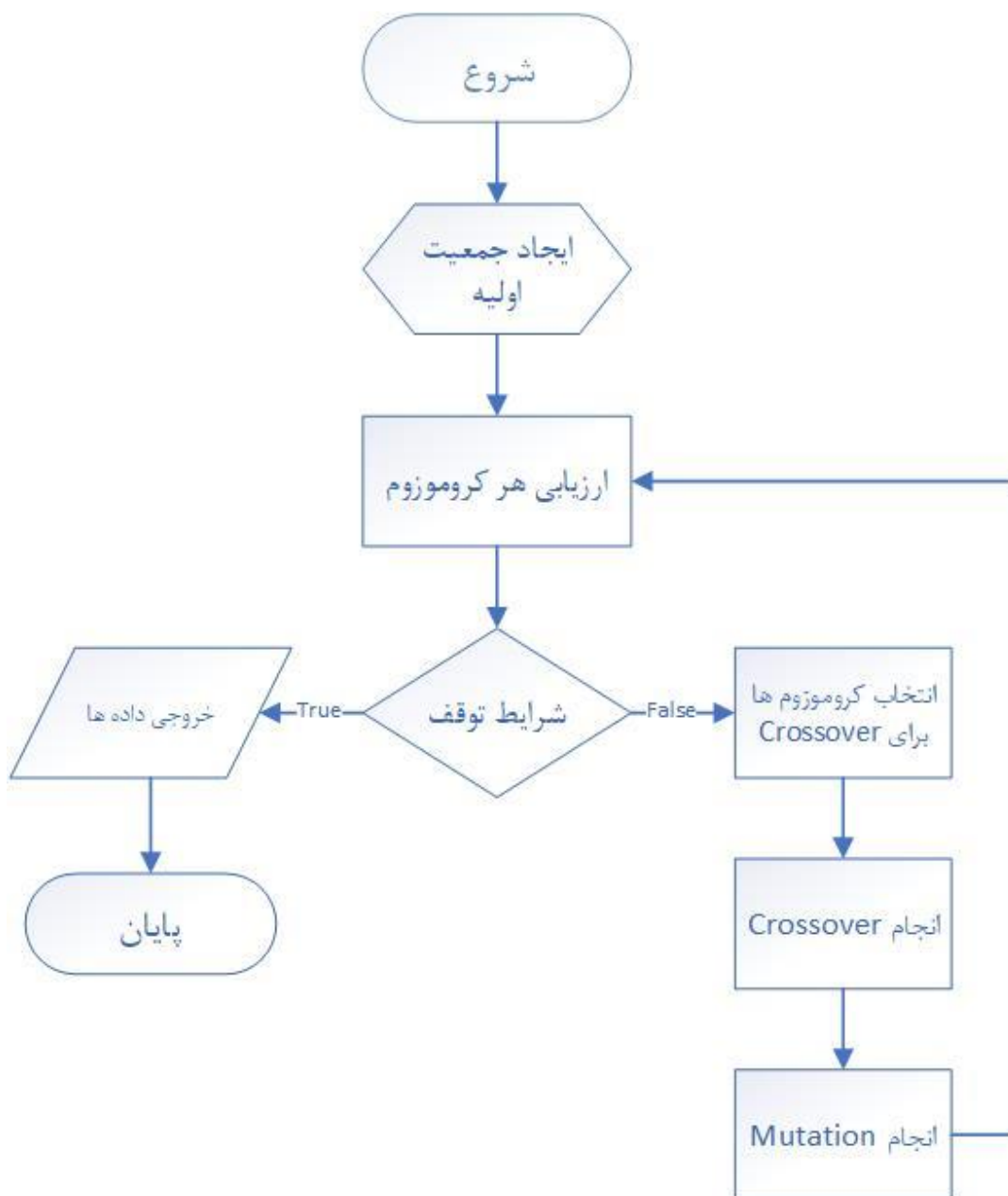
ارزیابی^۵، فرآیند محاسبه تناسب^۶ یک کروموزوم است. فیتنس معیار کیفیت کروموزوم نسبت به راه حل مورد نظر است. همچنین فرآیند با رسیدن به مقدار مشخصی فیتنس پایان می یابد. محاسبه فیتنس با ارزیابی هر کروموزوم به محدودیت های متوسط و نرم انتخاب شده به ویژه محل قرارگیری همه ی واحد ها^۷، استراحت ناهار انجام می شود.

^۴ adapt

^۵ Evaluation

^۶ Fitness

^۷ Subject Placement



فصل دوم: اجزای تشکیل دهنده سیستم:

سیستم از چهاربخش اصلی تشکیل شده است

- ۱- الگوریتم
- ۲- رابط کاربری
- ۳- دیتابیس
- ۴- مدیریت سناریو

الگوریتم

برای حل این مسئله راه حل متفاوتی وجود دارد یکی از این راه حل ها استفاده از الگوریتم های هوش جمعی مانند الگوریتم ژنتیک است.

در این پروژه به این دلیل از این الگوریتم استفاده شده است که الگوریتم ژنتیک قدرت و دوام بیشتری نسبت به سایر روش های مبتنی بر هوش مصنوعی دارد. بر خلاف سیستم های هوش مصنوعی قدیمی تر، الگوریتم ژنتیک با تغییر اندک مقادیر ورودی و یا با وجود مقادیر قابل توجهی از نویز در سیستم به راحتی قطع نمی شود. علاوه بر این، در جستجوی یک فضای حالت بزرگ، فضای حالت Multimodal و یا یک رویه چند بعدی، استفاده از الگوریتم ژنتیک مزیت های بسیار بیشتری نسبت به روش های جستجوی متداول در سایر تکنیک های بهینه سازی مانند برنامه ریزی خطی، جستجوی تصادفی و یا روش های جستجوی اول عمق، اول سطح و یا **praxis** دارد.

الگوریتم های ژنتیک برای حل مسئله، اصل بقای اصل را در میان اعضای یک جمعیت در طی نسل های متوالی شبیه سازی می کنند. هر نسل شامل جمعیتی از رشته کاراکتر ها است که مشابه کروموزوم هایی که در **DNA** ما دیده می شوند می باشند. هر فرد، نمایانگر یک نقطه در فضای جستجو و یک راه حل احتمالی خواهد بود. سپس اعضای هر نسل، در فرآیندی مشابه فرآیند تکامل موجودات زنده وارد می شوند.

الگوریتم های ژنتیک با استفاده از مبانی خاص، مشابه ساختار های ژنتیکی و رفتار کروموزوم ها در میان جمعیتی از افراد، عمل می کنند. این مبانی عبارتند از:

- اعضای یک جمعیت برای منابع و جفت گیری با یکدیگر رقابت می کنند.
- اعضای که در هر رقابت موفق تر هستند فرزندان بیشتری را نسبت به اعضای که عملکرد مناسبی نداشته اند ایجاد می کنند.

- ژن هایی از اعضای با عملکرد خوب در درون جمعیت منتشر می شود بنابراین، والدین خوب گاهی اوقات صاحب فرزندی می شوند که از هر کدام از والدین بهتر است.
- بنابراین، هر نسل متوالی برای زندگی در محیط اطراف خود مناسب تر خواهد بود.

رابط کاربری

برای ایجاد رابط کاربری در این پروژه از کتابخانه **PyQt5** استفاده شده است. **PyQt** یک **binding** (استفاده از کتابخانه های سی پلاس پلاس را در زبان پایتون ممکن می سازد) از فریم ورک **Qt** (بخوانید کیوت) می باشد که برای استفاده در زبان پایتون ایجاد شده است. خود **Qt** در حقیقت مجموعه ای از کتابخانه ها و ابزارهای توسعه ی مستقل از پلتفرم در زبان سی پلاس پلاس است **PyQt** . شامل انتزاع از مفاهیم رابط گرافیکی و کتابخانه های مخصوص شبکه، پردازش موازی، عبارت های قاعده دار، پایگاه داده **SQL** و... می باشد.

این کتابخانه قدرتمند می تواند تا حد بسیار زیادی در ایجاد پروژه های گرافیکی کمک می کند. این کتابخانه قدرتمند در حقیقت مانند یک چاقوی همه کاره است. تقریباً می تواند هر برنامه دلخواهی را با استفاده از کتابخانه **pyqt5** تولید کرد. در ابتدا بسیاری زبان برنامه نویسی پایتون را چندان جدی نمی گرفتند اما با روی کار آمدن کتابخانه های قدرتمند این زبان پیشرفت های جالبی ایجاد شد **PyQt** . که توسط **Riverbank Computing** ایجاد شده است، یک نرم افزار رایگان (دارای مجوز **GPL**) است و از سال ۱۹۹۹ در حال توسعه است **PyQt5** . در سال ۲۰۱۶ منتشر شد و آخرین بار در اکتبر ۲۰۲۱ به روز شد. هدف از انتخاب این ابزار برای ایجاد رابط کاربری، سبکی و درعین حال قدرتمندی این کتابخانه است که ما را قادر می سازد تا برای سیستم عامل های مختلف بتوانیم رابط کاربری بسازیم. یا به بیان دقیق تر برنامه **cross-platform** ایجاد کنیم

دیتابیس

برای ذخیره سازی و ساماندهی داده های اولیه و همچنین نتایج ایجاد شده نرم افزار باید از یک دیتابیس استفاده کنیم. در این پروژه به دلایل زیر از دیتابیس **SQLite** استفاده شده است:

- تمرکز اصلی **SQLite** بر ارائه ی بانک داده ی قدرتمند سازگار با **SQL**، بدون وابستگی به موارد اضافه است. همان طور که از نام این پایگاه داده نیز برمی آید، می توان آن را یک راهکار سبک محسوب کرد که تقریباً روی هر بستری که از **C** و ذخیره سازی فایل پشتیبانی می کند، اجرا می شود. امکان ارتباط بین بانک داده ی **SQLite** با زبان های برنامه نویسی سطح بالا در دسترس توسعه دهندگان قرار دارد.

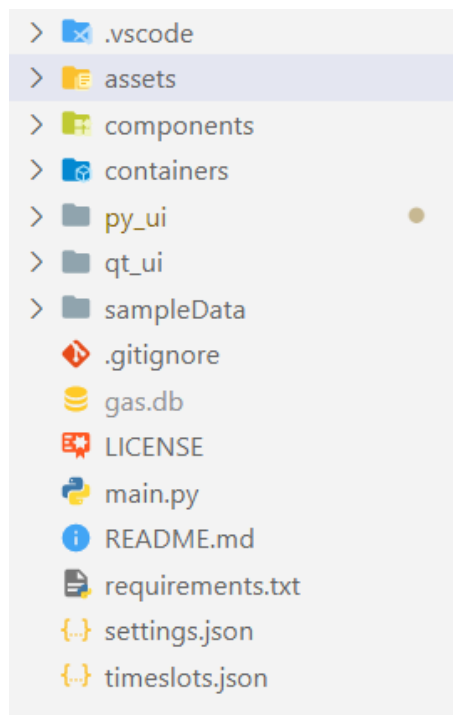
- از آنجا که بانک داده‌ی SQLite، در واقع یک فایل ساده است، بنابراین قابلیت حمل بسیار بالایی دارد و می‌توان به راحتی نسخه‌ی پشتیبان آن را ایجاد کرد. عدم نیاز به بخش سرور باعث شده است راه‌اندازی SQLite و استفاده از آن بسیار ساده باشد. شما می‌توانید حتی بدون راه‌اندازی محیط توسعه‌ی کامل، از مزایای SQLite در پروژه‌های خود بهره ببرید. علاوه بر این، استفاده از بانک داده به طی کردن فرآیندهای طولانی، راه‌اندازی مجدد یا بررسی مسائل امنیتی نیاز ندارد.
- استفاده از SQLite در برنامه‌های کاربردی باعث افزایش انعطاف‌پذیری و کاهش زمان توسعه خواهد شد. بهره گرفتن از فایل‌های متنی ساده به عنوان بانک داده‌ی SQLite برای پیکربندی، ذخیره‌سازی و دسترسی به داده‌ها در دستگاه‌های مختلف یکسان است و همین مورد به حفظ عملکرد ثابت این فناوری کمک می‌کند.
- بانک‌های داده معمولاً از سیستم‌هایی برای محافظت در برابر آسیب‌های احتمالی بهره می‌برند که این ویژگی در فایل‌های معمولی دیده نمی‌شود. SQLite، یک بانک داده‌ی تراکنشی (transactional) است، بنابراین می‌توانید از رویدادهای موفق جزئی اجتناب کنید. اگر یک عملیات در یک تراکنش با شکست مواجه شود، عملیات موفق نیز بازگردانی می‌شود و پایگاه داده را به حالت اولیه‌ی خود باز می‌گرداند.
- SQLite در برابر خطاهای ذخیره‌سازی و سناریوهایی که به دلیل کمبود منابع رم رخ می‌دهد، آسیب‌پذیر نیست. امکان بازیابی بانک‌های داده در صورت بروز مشکل در سیستم یا قطعی برق وجود دارد که این اقدام به ایمنی داده‌ها کمک می‌کند. پایگاه کد توسط مجموعه‌ی آزمایشی بسیار فراگیری با پوشش ۱۰۰ درصد، داده‌ها را تحت مدیریت خود قرار می‌دهد.

مدیریت سناریو

مدیریت سناریو در واقع بخش‌های مختلف پروژه شامل دیتابیس، و رابط کاربری را به یکدیگر متصل می‌سازد و همچنین thread ها را مدیریت می‌کند. تا نتیجه دلخواه ایجاد شود.

فصل سوم: ساختار پروژه

ساختار کلی پروژه را در شکل ۳-۱ مشاهده می کنید.



شکل ۳-۱

دایرکتوری Components

این دایرکتوری شامل ماژول ها و کلاس های اصلی پروژه است که در ادامه هرکدام را به تفصیل توضیح خواهیم داد. تمام ماژول های پروژه به صورت شیء گرا و فانکشنال طراحی شده اند تا روند طراحی و توسعه در آینده راحت تر باشد.

دایرکتوری Containers

این دایرکتوری شامل فانکشن های مربوط به GUI می باشد. فایل مربوط به هر پنجره جدا می باشد که با نام کلاس اصلی آن مشخص شده است

دایرکتوری py_ui

کتابخانه PyQt از روی فایل های این دایرکتوری GUI را می سازد به عبارت بهتر فایل های این دایرکتوری ساختار رابط کاربری را مشخص می کنند.

فایل **gas.db**

فایل دیتابیس SQLite اصلی پروژه که داده های اولیه شامل مشخصات اساتید، دروس، کلاس ها (اتاق ها) و نیز نتایج حاصل از الگوریتم ژنتیک در آن ذخیره شده است.

فایل **requirements.txt**

پکیج ها و کتابخانه های مورد استفاده در پروژه در این فایل نوشته شده اند که به راحتی با استفاده از دستور `pip install -r requirements.txt` قابل نصب باشند

فایل **settings.json**

تنظیمات کلی برنامه و نیز تنظیمات مربوط به مراحل مختلف الگوریتم ژنتیک بصورت فرمت **json** در این فایل نوشته شده اند تا از طریق الگوریتم و نیز رابط کاربری قابل دسترسی و تغییر باشند و همچنین با بسته شدن نرم افزار به حالت اولیه بازگردند.

فایل **timeslots.json**

تعداد و نام بازه های زمانی در این فایل به فرم **json** قرار گرفته اند که تعداد و نام آنها با تغییر این فایل امکا پذیر خواهد بود.

فصل چهارم : ساختار ساخت زمانبندی

در این فصل با اجزای مختلف الگوریتم ژنتیک و تکنیک استفاده شده در آن آشنا خواهیم شد و کدهای این قسمت را بررسی خواهیم کرد. کلاس ها و توابع مربوط به الگوریتم ژنتیک در فایل `GeneticAlgoritim.py` واقع در دایرکتوری `components` نوشته شده اند.

کتابخانه های مورد نیاز الگوریتم

```
from PyQt5 import QtCore
from components import Settings,Utilities
from operator import itemgetter
from collections import Counter
import copy
import itertools
import numpy as np
```

خط اول: کلاس `QtCore` برای کار با `thread` ها در این ماژول استفاده شده است تا الگوریتم بدون ایجاد اختلال در عملکرد رابط کاربری بتواند به عنوان یک `Thread` پردازشی مجازی به کار خود ادامه دهد.

خط دوم: فایل تنظیمات و توابع کمکی که در فایل `utilities.py` قرار گرفته اند را فراخوانی می کند.

خط سوم: تابع `Itemgetter` را از کتابخانه `operator` فراخوانی میکند که در این ماژول برای مرتب سازی (`sort`) های جهت دار لیست داده ها مورد استفاده قرار می گیرد.

خط چهارم: تابع `Counter` را از کتابخانه `collections` فراخوانی می کند. این تابع مقدار هر کاراکتر در یک رشته یا مقدار هر عضو یک لیست را می شمارد و نتیجه را بصورت یک دیکشنری برمیگرداند که برای محاسبه های آماری جمعیت در این ماژول استفاده شده است

خط پنجم: کتابخانه `copy` را فراخوانی می کند. به بیان ساده این کتابخانه برای ایجاد کپی از یک لیست موجود در حافظه بدون استفاده از اشاره گرها استفاده می شود. و توابع مختلفی دارد.

خط هفتم: `NumPy` که مخفف `Numerical Python` است، برای محاسبات عددی متنوعی در پایتون به کار می رود. محاسبات به کمک آرایه ها در `NumPy` سرعت خوبی دارند و علاوه بر آن، توابع این پکیج در ساخت پکیج های محاسباتی دیگر مورد استفاده قرار گرفته است.

کلاس کروموزم در الگوریتم ژنتیک

```
class Chromosome:
    # data = {
    #     sections: {
    #         id: {
    #             details: {
    #                 subject: [roomId,
    #                 instructorId,
    #                 [day / s],
    #                 startingTS,
    #                 length
    #             ]
    #         },
    #         schedule: [days]
    #     },
    #     instructors && rooms: {
    #         id: [
    #             [days] // Timeslots
    #             [1, None, 1, None, 1, False] // Example
    #             None = Vacant, False = Unavailable
    #         ]
    #     },
    #     unplaced: {
    #         'sections': {
    #             id: [] // Section ID and unplaced subjects
    #         }
    #     }
    # }
```

ساختار دیتای یک کروموزم در کامنت کد بالا مشاهده می کنید. همانطور که مشاهده می کنید در یک کروموزم تمام اطلاعات اولیه و همچنین یک شیوه از چیدمان ذخیره خواهد شد. تا در صورت تغییر اطلاعات سایر کروموزم دستخوش تغییر نشود. درست مشابه کروموزم انسان هر سلول (کروموزم) حاوی تمام اطلاعات (ژنهای) آن فرد خواهد بود.

```

def __init__(self, data):
    self.fitness = 0
    self.fitnessDetails = []
    self.data = {
        'sections': {},
        'instructors': {},
        'rooms': {},
        'unplaced': {
            'sections': {}
        }
    }
    self.rawData = data
    self.settings = Settings.getSettings()
    self.buildChromosome()
    self.launchBreak = self.settings['lunchbreak']

```

تابع سازنده این کلاس شامل فیتنس و اطلاعات مربوط به فیتنس کروموزوم و اطلاعات خام (**rawData**) بصورت یک ورودی از کلاس دریافت می شوند و تغییراتی بر روی آن صورت خواهد گرفت و پس از تغییرات در متغیر **data** ذخیره خواهند شد. همچنین تنظیمات که با استفاده از تابع **getSettings** که از ماژول **Settings** فراخوانی خواهد شد. داخل هر کروموزوم فراخوانی خواهد شد. تا در موقعیت های مختلف درون هر آبجکت کلاس کروموزوم قابل دستیابی باشد.

```

def buildChromosome(self):
    rawData = self.rawData
    # {id: {details: [subject: []], schedule: [days]}}
    sections = rawData['sections']
    for section in sections:
        self.data['sections'][section] = {'details': {}, 'schedule': []}
        self.data['sections'][section]['details'] = {key: [] for key in
sections[section][2]}
        sectionTimetable = []
        for timeslotRow in sections[section][1]:
            sectionTimetable.append([None if day == 'Available' else False
for day in timeslotRow])
        self.data['sections'][section]['schedule'] = sectionTimetable
        self.data['unplaced']['sections'][section] = []
    # {id: [days]}
    instructors = rawData['instructors']
    for instructor in instructors:
        instructorTimetable = []
        for timeslotRow in instructors[instructor][2]:
            instructorTimetable.append([None if day == 'Available' else
False for day in timeslotRow])
        self.data['instructors'][instructor] = instructorTimetable
    # {id: [days]}
    rooms = rawData['rooms']
    for room in rooms:
        roomTimetable = []
        for timeslotRow in rooms[room][2]:
            roomTimetable.append([None if day == 'Available' else False
for day in timeslotRow])
        self.data['rooms'][room] = roomTimetable

```

تابع `buildChromosome` که در سازنده کلاس `Chromosome` فراخوانی شده است. یک کروموزم جدید ایجاد می کند. در این تابع اطلاعات خام (`rawData`) که تقریباً بصورت دست نخورده از پایگاه داده نرم افزار واکنشی شده بودند دستخوش تغییراتی می شوند تا برای پردازش های بعدی آماده شوند. (Preprocessing)

```

# [roomId, [sectionId], subjectId, instructorID, [day/s], startingTS, length]
def insertSchedule(self, schedule):
    # Validate schedule details
    isValid = self.validateSchedule(copy.deepcopy(schedule))
    if isValid is not True:
        return isValid
    data = self.data
    # [roomId, instructorId, [day/s], startingTS, length]
    subjectDetails = [schedule[0], schedule[3], schedule[4], schedule[5],
schedule[6]]
    # Insert details into section data
    for section in schedule[1]:
        data['sections'][section]['details'][schedule[2]] = subjectDetails
    # Update instructor and room timetable
    for timeslot in range(schedule[5], schedule[5] + schedule[6]):
        for day in schedule[4]:
            if schedule[3]:
                data['instructors'][schedule[3]][timeslot][day] =
schedule[1]
                data['rooms'][schedule[0]][timeslot][day] = schedule[1]
    # False signifies no error in insertion
    return False

```

تابع `insertSchedule` یک لیست بصورت زیر دریافت می کند. و سپس بررسی می کند که آیا از نظر معیارهای طراح سیستم این زمانبندی قابل جایگذاری در برنامه هفتگی می باشد یا خیر. به عبارت دقیق تر زمانبندی ورودی را بررسی می کند تا با دیگر معیار های زمانبندی تداخل نداشته باشد.

اگر زمانبندی درس وارد شده با یکی از معیار ها ناسازگار باشد تابع یک ارور از ۱ تا ۴ برمی گرداند که مشخص می کند که این زمانبندی با کدام یک از معیار ها در تداخل بوده تا واکنش مناسب نسبت به آن صورت گیرد.

در غیر اینصورت زمانبندی در دیکشنری اتاق وارد خواهد شد که نشان می دهد که یک کلاس مشخص در یک روز و یک زمان مشخص چه کلاسی دارد. همین رویداد برای دیکشنری استاد (`instructor`) نیز انجام خواهد شد. و در نهایت `False` نیز بازگردانده خواهد شد.

```
def validateSchedule(self, schedule):
    if not self.isRoomTimeslotAvailable(schedule):
        return 1
    if not self.isInstructorTimeslotAvailable(schedule):
        return 2
    if not self.isSectionTimeslotAvailable(schedule):
        return 3
    if self.launchBreak:
        if not self.isLunchTime(schedule):
            return 4
    return True
```

تابع `validateSchedule` نیز که در تابع `insertSchedule` مورد استفاده قرار گرفته بود، صحت زمانبندی `(schedule)` مورد نظر را بررسی می کند.

چهار معیار برای هر زمانبندی در نظر گرفته شده است:

۱. `isRoomTimeslotAvailable`: بررسی می کند که اتاق در زمان مورد نظر خالی باشد و با کلاس دیگر تداخل نداشته باشد

۲. `isInstructorTimeslotAvailable`: بررسی می کند که استاد در زمان مورد نظر تایم خالی داشته باشد و کلاس دیگری در آن زمان برای استاد تعریف نشده باشد و یا استاد در آن زمان در دانشگاه حضور داشته باشد.

۳. `isSectionTimeSlotAvailable`: بررسی می کند که گروه درسی در آن زمان اجازه برگزاری کلاس داشته باشد

۴. `isLaunchTime`: بررسی می کند که کلاس در زمان ناهار و نماز برگزار نشود.

```
# we shouldn't have any Class in [12:40, 13:30]
def isLunchTime(self, schedule):
    subject_timeslot = [timeslots for timeslots in range(schedule[5],
schedule[5] + schedule[6])] # Create list of busy timeslots
    if 6 in subject_timeslot: # if [6,7,8] or [5,6] or ...
        return False
    return True
```



```

def isRoomTimeslotAvailable(self, schedule):
    room = self.data['rooms'][schedule[0]]
    for timeslotRow in range(schedule[5], schedule[5] + schedule[6]):
        for day in schedule[4]:
            if room[timeslotRow][day] is not None:
                return False
    return True

def isSectionTimeslotAvailable(self, schedule):
    rooms = self.data['rooms']
    sections = self.data['sections']
    # Check for each room if on the given subject range, the section has
class
    for room in rooms:
        for timeslotRow in range(schedule[5], schedule[5] + schedule[6]):
            for day in schedule[4]:
                roomDayTimeslot = rooms[room][timeslotRow][day]
                # Check if timeslot is blank
                if roomDayTimeslot is None:
                    continue
                # Check if section is in timeslot
                # for section in schedule[1]:
                #     if section in roomDayTimeslot:
                #         return False
            # Check for section unavailable times
        for section in schedule[1]:
            for timeslotRow in range(schedule[5], schedule[5] + schedule[6]):
                for day in schedule[4]:
                    if sections[section]['schedule'][timeslotRow][day] is not
None:
                        return False
    return True

def isInstructorTimeslotAvailable(self, schedule):
    # Pass if no instructor is set
    if not schedule[3]:
        return True
    instructor = self.data['instructors'][schedule[3]]
    for timeslotRow in range(schedule[5], schedule[5] + schedule[6]):
        for day in schedule[4]:
            if instructor[timeslotRow][day] is not None:
                return False
    # Check if instructor can still teach
    maxLoad = self.rawData['instructors'][schedule[3]][1] * 2
    for timeslotRow in instructor:
        for day in timeslotRow:
            if day:
                maxLoad -= 1

```

کلاس GeneticAlgorithm

```
class GeneticAlgorithm(QtCore.QThread):  
    # Current phase of the algorithm  
    statusSignal = QtCore.pyqtSignal(str)  
    # Genetic algorithm variable details  
    detailsSignal = QtCore.pyqtSignal(list)  
    # Running process type  
    operationSignal = QtCore.pyqtSignal(int)  
    # List of chromosomes for preview  
    dataSignal = QtCore.pyqtSignal(list)  
    # Operation ProgressBar  
    progressBarSignal = QtCore.pyqtSignal(int)  
    # Operation Progress Status  
    progressSignal = QtCore.pyqtSignal(str)
```

کلاس GeneticAlgorithm، کلاس اصلی اجرای الگوریتم ژنتیک است که الگوریتم را مرحله به مرحله تا رسیدن به جواب ایده آل اجرا می کند. قبل از فرخوانی تابع سازنده، سیگنال های GUI فراخوانی شده اند. اطلاعات از الگوریتم به صورت سیگنال همگام به GUI منتقل می شوند تا اطلاعات پردازشی و مرحله و روند اجرای الگوریتم را به رویت کاربر برسانند.

```

def __init__(self, data):
    self.averageFitness = 0
    self.pastAverageFitness = 0
    self.running = True
    self.chromosomes = []
    self.data = {
        'rooms': [],
        'instructors': [],
        'sections': [],
        'subjects': []
    }
    self.tournamentSize = .04
    self.elitePercent = .05
    self.lowVariety = 55
    self.highestFitness = 0
    self.lowestFitness = 100
    self.elites = []
    self.matingPool = []
    self.offsprings = []
    self.tempChromosome = None
    self.tempSections = None
    self.data = data
    self.settings = Settings.getSettings()
    self.stopWhenMaxFitnessAt = self.settings['maximum_fitness']
    self.mutationRate = self.settings['mutation_rate_base']
    self.mutationRateStep = self.settings['mutation_rate_step']
    super().__init__()

```

سازنده این کلاس شامل متغیر هایی است که در ادامه آنها را معرفی خواهیم کرد. متغیر **averageFitness** که مقدار فیتنس میانگین کروموزوم های جمعیت فعلی را ذخیره می کند. و متغیر **pastAverageFitness** مقدار فیتنس میانگین جمعیت قبل را ذخیره می کند تا بتوان این دو مقدار را با یکدیگر مقایسه کرد. متغیر **running** که در حال اجرا بودن الگوریتم را نشان می دهد. لیست **chromosomes** کروموزوم های جمعیت فعلی را در خود ذخیره می کند.

مقدار **tournamentSize** برابر است با درصدی از جمعیت فعلی که در الگوریتم **tournamentSelection** با یکدیگر برای نسل بعدی رقابت می کنند.

مقدار **elitePercent** برابر است با درصدی از جمعیت که با استفاده از الگوریتم **eliteSelection** مستقیماً به نسل بعد منتقل می شوند.

لیست **elites** که جمعیت نخبه یا برتر را در خود نگه می دارد.

لیست **matingPool** که زوج هایی از کروموزوم های که قرار است با یکدیگر **crossover** داده و تشکیل **offspring** دهند را در خود نگه داشته و **offspring** ها در یک لیست **offsprings** ذخیره خواهند شد.

متغیر **mutationRate** مقدار فعلی نرخ جهش در جمعیت و متغیر **mutationRateStep** نرخ تغییر جهش در جمعیت را در خود نگه می دارند.

```

def initialization(self):
    # Generate population based on minimum population
    self.generateChromosome(self.settings['minimum_population'])

    def generateChromosome(self, quantity):
        self.progressBarSignal.emit(0)
        for i in range(quantity):
            self.statusSignal.emit('ایجاد {} از {} کروموزوم'.format(i, quantity)) #
Display Chromosome creation
            self.progressBarSignal.emit(int(i * 100 / quantity))
            self.tempChromosome = Chromosome(self.data) # Create new
Chromosome
            # {id: [[subjectIds](roomId)]}
            self.tempSections = sections = {key: [value[2], value[3]] for
(key, value) in
                                copy.deepcopy(self.data['sections']
)].items()}
            # [roomId]
            self.rooms = rooms = list(self.data['rooms'].keys()) # Get all
roomId

            self.generateSubjectPlacementsForSections(sections)
            self.chromosomes.append(self.tempChromosome)

```

تابع `generateChromosome` به اندازه ورودی `quantity` کروموزوم ایجاد می کند و با فراخوانی تابع `generateSubjectPlacementsForSections` برای هر گروه یک چیدمان ارائه می دهد (در بخش بعدی با عملکرد این متد بیشتر آشنا خواهیم شد). در نهایت کروموزوم ایجاد شده در لیست کروموزم های جمعیت ذخیره خواهد شد.

```

# {id: [[subjectIds](roomId)}
def generateSubjectPlacementsForSections(self, sections):
    # Maximum length of section subjects
    maxSubjects = max(len(subjects[0]) for subjects in sections.values())
    # Put one random section subject per turn
    for i in range(maxSubjects):
        for section in sections:
            subjectList = sections[section][0]
            if not len(subjectList):
                continue
            subjectToPlace = np.random.randint(0, len(subjectList))
            result = self.generateSubjectPlacement([section],
            subjectList[subjectToPlace])
            if not result:
                self.tempChromosome.data['unplaced']['sections'][section].
            append(subjectList[subjectToPlace])
            sections[section][0].pop(subjectToPlace)

```

این متد با اجرای متد **generateSubjectPlacement** برای هر گروه (Subject) سعی در این دارد که برای دروس موجود در هر گروه یک زمانبندی ایده آل ارائه دهد. اگر این چیدمان بدون ارور انجام شود **False** را برمی گرداند ولی اگر این چیدمان همراه با ارور باشد درسی که چیده نشده در لیست **unplacement** قرار خواهد گرفت.

تابع **generateSubjectPlacement** تا زمانی که متغیر **generating** برابر **True** باشد تلاش می کند تا یک زمانبندی مناسب برای درس (subject) وارد شده پیدا کند. اگر انتخاب زمانبندی ارور داشته باشد سعی در رفع ارور با انتخاب دوباره اتاق، استاد و یا تایم دارد. در غیر اینصورت با استفاده از متد **insertSchedule** از آبجکت کروموزوم مورد نظر، زمانبندی جدیدی برای درس انتخابی ارائه می دهد.

متد selectRoom

```
def selectRoom(self, subject):
    room = None
    while not room:
        candidate = np.random.choice(self.rooms)
        if self.data['subjects'][subject][6] ==
self.data['rooms'][candidate][1]:
            room = candidate
    return room
```

متد selectRoom یک اتاق بصورت تصادفی از لیست اتاق های مجاز برای درس (subject) مورد نظر، انتخاب می کند.

متد selectInstructor

```
def selectInstructor(self, subject):
    instructor = None
    subjectInstructors = self.data['subjects'][subject][4]
    while not instructor and len(subjectInstructors):
        instructor = np.random.choice(subjectInstructors)
    return instructor
```

متد selectInstructor یک استاد بصورت تصادفی از لیست اساتیدی که درس (subject) مورد نظر را تدریس می

کنند، انتخاب می کنند.

متد selectTimeDetails

```
def selectTimeDetails(self, subject):
    days = [0, 1, 2, 3, 4, 5]
    np.random.shuffle(days)
    hours = self.data['subjects'][subject][1]
    # Check if hours can be splitted with minimum session of 1 hour or 2
    timeslot
    selected_day = [np.random.randint(0, 6)]
    # To convert hours into timetable timeslots
    # hours = hours / .5
    startingTimeslot_status = False
    # Starting slot selection
    startingTime = self.settings['starting_time']
    endingTime = self.settings['ending_time']
    while not startingTimeslot_status:
        candidate = np.random.randint(startingTime, endingTime -
startingTime)
        # Validate if subject will not overpass operation time
        if (candidate + hours) <= endingTime:
            startingTimeslot = candidate
            startingTimeslot_status = True
    return [selected_day, int(startingTimeslot), int(hours)]
```

متد selectTimeDetails در ابتدا یک روز بصورت کاملاً تصادفی انتخاب می کند. سپس یک تایم اسلات از بین تایم

اسلات های موجود در روز یکی را بصورت تصادفی انتخاب می کند.


```

def evaluate(self):
    totalChromosomeFitness = 0
    self.pastAverageFitness = copy.deepcopy(self.averageFitness)
    self.lowestFitness = 100
    self.highestFitness = 0
    self.progressBarSignal.emit(0)
    for index, chromosome in enumerate(self.chromosomes): # For each
        chromosome
            self.statusSignal.emit('ارزیابی کروموزم #{} از {}'.format(index + 1,
len(self.chromosomes)))
            self.progressBarSignal.emit(int((index + 1) * 100 /
len(self.chromosomes)))
            chromosome.fitness = self.evaluateAll(chromosome)
            totalChromosomeFitness += chromosome.fitness # Add chromosome
fitness to total fitness
            self.averageFitness = totalChromosomeFitness /
len(self.chromosomes)
            self.highestFitness = chromosome.fitness if chromosome.fitness >
self.highestFitness else self.highestFitness # Update Highest Fitness
            self.lowestFitness = chromosome.fitness if chromosome.fitness <
self.lowestFitness else self.lowestFitness # Update Lowest Fitness
            chromosomeFitness = sorted(enumerate(map(lambda chromosome:
chromosome.fitness, self.chromosomes)),
key=itemgetter(1)) # Sort chromosomes by
fitness
            # Emit top five chromosomes
            self.dataSignal.emit(
                list(map(lambda chromosome: [self.chromosomes[chromosome[0]],
chromosome[1]], chromosomeFitness[-5:])))

```

محاسبه فیتنس تا حد زیادی بر اساس مقدار موضوع ترسیم شده در مقایسه با موضوعات مورد نیاز است. با این حال، محاسبه همچنان به ماتریس ارزیابی ارائه شده توسط کاربر بستگی دارد. ماتریس ارزیابی مجموعه ای از وزن های محدودیت است که قابلیت شکل دادن به جواب ها را دارد. این فهرستی از اولویت بندی محدودیت ها با استفاده از توزیع صد درصد است.

```

# Evaluation weight depends on settings
def evaluateAll(self, chromosome):
    matrix = self.settings['evaluation_matrix'] # Get evaluation matrix
    # If subject placement is enabled else 0
    subjectPlacement = self.evaluateSubjectPlacements(chromosome) if
matrix['subject_placement'] !=0 else 0
    # If idle time is enabled else 0
    idleTime = self.evaluateInstructorIdleTime(chromosome) if
matrix['idle_time'] !=0 else 0

    chromosome.fitnessDetails = copy.deepcopy([subjectPlacement,
idleTime])
    return (
        (subjectPlacement * matrix['subject_placement'] / 100) +
        (idleTime * matrix['idle_time'] / 100)
    )

```

| Name | Function |
|-------------------|--|
| Subject Placement | تلاش برای ایجاد برنامه زمانی معتبر برای همه واحدهای همه بخش ها |
| idleTime | تلاش برای به حداقل رساندن تایم های خالی اساتید |

برای محاسبه فیتنس کل از فرمول زیر استفاده می شود:

X برابر هر کروموزم انتخابی است:

$$\begin{aligned}
 fitness(x) = & (g(x) * gW) + (h(x) * hW) + (i(x) * iW) + (j(x) * jW) + (k(x) * kW) \\
 & + (l(x) * lW) + (m(x) * mW)
 \end{aligned}$$

Subject Placement

این مرحله ارزیابی فیتنس پایه کروموزوم را بر اساس ایجاد برنامه زمانی معتبر برای همه واحدهای همه ی بخش ها محاسبه می کند.

$$fitness = \frac{\text{تعداد دروسی که دارای زمانبندی معتبر هستند}}{\text{تعداد کل دروس}}$$

idleTime

این مرحله از محاسبه فیتنس کروموزم براساس تلاش الگوریتم برای به حداقل رساندن تایم های خالی بین دروس یک روز اساتید ارزیابی میشود

$$fitness = \frac{\text{تعداد واحدهای زمانی پیوسته که بین دو درس نباشند}}{\text{تعداد کل واحدهای زمانی یک روز}}$$

```

def adapt(self):
    deviation = self.getFitnessDeviation() # it returns sigma & sigma instances
    self.alignPopulation(deviation[0], deviation[1])
    self.adjustMutationRate()

# Function to find Mean Deviation of fitnesses
# sigma = [sigma], sigmaInstances = {sigma: instance%}
def getFitnessDeviation(self):
    populationCount = len(self.chromosomes) # find number of population
    fitnesses = [chromosome.fitness for chromosome in self.chromosomes]
    mean = np.mean(fitnesses)
    sigmas = [int(fitness - mean) for fitness in fitnesses]
    sigmaInstances = {sigma: (instance / populationCount) * 100 for sigma, instance in
                      dict(Counter(sigmas)).items()}
    return [sigmas, sigmaInstances]

def alignPopulation(self, sigmas, sigmaInstances):
    populationCount = len(self.chromosomes) # find number of population
    sigmaStartingInstance = list(sigmaInstances.values())[0] # get first sigmaInstance value
    if sigmaStartingInstance > self.lowVariety: #TODO: Check lowVariety
        # Add the excess percentage of instances on first sigma to population
        generate = int((int(sigmaStartingInstance - self.lowVariety) / 100) * populationCount)
        while generate + populationCount > self.settings['maximum_population']:
            generate -= 1
        self.generateChromosome(generate)
    else:
        # Remove the excess percentage of instances on first sigma to population
        sortedSigmas = sorted(enumerate(sigmas), key=itemgetter(1))
        remove = int((int(self.lowVariety - sigmaStartingInstance) / 100) * populationCount)
        while populationCount - remove < self.settings['minimum_population']:
            remove -= 1
        remove = [sortedSigmas[index][0] for index in range(remove)]
        self.chromosomes = [chromosome for index, chromosome in enumerate(self.chromosomes) if index not in
remove]

```

جمعیت اصطلاحی است برای گروهی از راه حل ها. قبل از تولید راه حل، اپراتور با استفاده از سیستم می تواند تنظیمات اجرا الگوریتم را تغییر دهد. تنظیمات "حداقل جمعیت"^{۱۰} و "حداکثر جمعیت"^{۱۱} حد تغییر جمعیت را مشخص می کند. محاسبه تراز جمعیت که نشان می دهد که چه مقدار تغییر در جمعیت لازم است با استفاده از یک فرمول زیر انجام می شود. در واقع در این تابع الگوریتم تصمیم می گیرد جمعیت جدید (مهاجر) به جمعیت وارد کند یا خیر

$$change(x) = \frac{\left(\frac{\sum instances\ of\ x\ with\ (-1 < deviation < 1)}{population\ count} * 100 \right) - low\ variety\ Setting}{100} * population\ count$$

تنظیم نرخ جهش^{۱۲}

نرخ جهش شانس برای هر کروموزوم برای ایجاد تغییرات تصادفی است. تنظیم نرخ جهش زمانی اتفاق می افتد که محرکی که قبل از اجرا تنظیم شده است، برآورده شود. هنگامی که یک نسل بدون تنظیم نرخ جهش کامل می شود، میزان جهش ۰.۵٪ کاهش می یابد، در غیر این صورت ۰.۵٪ افزایش می یابد. محرک تنظیم نرخ جهش^{۱۳} از فرمول زیر محاسبه می شود:

$$change = 0.5 * (-1\ if\ average\ fitness - previous\ average\ fitness < adjustment\ trigger\ setting\ else\ 1)$$

^{۱۰} Minimum Population

^{۱۱} Maximum Population

^{۱۲} Mutation Rate Adjustment

^{۱۳} mutation rate adjustment trigger

انتخاب جمعیت هدف^{۱۴}

پس از ارزیابی، کروموزوم ها برای شرکت در تولید مثل انتخاب می شوند. احتمال انتخاب کروموزوم هایی با فیتنس بالاتر بیشتر است. اینجاست که بقای شایسته ترین ها فرا می رسد.

```
def selection(self):
    population = len(self.chromosomes)
    # Get All Chromosome Fitnesses
    chromosomeFitness = [self.chromosomes[chromosome].fitness for chromosome in
range(len(self.chromosomes))]
    # Select number of elites that will ensure there will be even offspring to be generated
    eliteCount = round(population * self.elitePercent) # Calculate Elite population
    if population % 2 == 0:
        eliteCount = eliteCount if eliteCount % 2 == 0 else eliteCount + 1
    else:
        eliteCount = eliteCount if eliteCount % 2 != 0 else eliteCount + 1
    self.progressBarSignal.emit(100)
    self.statusSignal.emit('انتخاب {} Elites'.format(eliteCount))
    sortedFitness = sorted(enumerate(chromosomeFitness), key=itemgetter(1))
    elites = list(map(lambda chromosome: chromosome[0], sortedFitness[eliteCount * -1:]))
    matingPool = []
    matingPoolSize = int((population - eliteCount) / 2)
    tournamentSize = int(self.tournamentSize * population)
    if tournamentSize > 25:
        tournamentSize = 25
    # Fill mating pool with couples selected by multiple tournaments
    self.progressBarSignal.emit(0)
    for i in range(matingPoolSize):
        self.statusSignal.emit('ایجاد # {} از {} زوج'.format(i + 1, matingPoolSize))
        self.progressBarSignal.emit(int((i / matingPoolSize) * 100))
        couple = []
        while len(couple) != 2:
            winner = self.createTournament(tournamentSize, chromosomeFitness)
            if winner not in couple:
                couple.append(winner)
        matingPool.append(couple)
    self.elites = elites
    self.matingPool = matingPool
```

روش‌های مختلفی برای انتخاب کروموزوم‌ها وجود دارد که همگی مزایا و معایب خود را دارند. داشتن مناسب‌ترین نوع انتخاب^{۱۵} به جلوگیری از همگرایی اولیه و ترویج راه حل‌های متنوع (ایجاد تنوع در جمعیت) کمک می‌کند. به عنوان یک راه حل شایسته‌گرایانه از الگوریتم ژنتیک، n درصد بالای جمعیت انتخاب می‌شود که در نسل بعدی با همان ژن‌ها (بدون تغییر) پیشروی کنند.

بقیه مراحل انتخاب قرار است با اجرای مسابقات^{۱۶} متعدد بر روی جمعیت انجام شود. شرکت کنندگان در مسابقات به صورت تصادفی با تعداد k درصد از جمعیت موجود انتخاب می‌شوند اما سقف آن بیست و پنج (۲۵) است. مسابقات تا زمانی که جفت‌های کافی انتخاب شوند ادامه خواهد داشت. مثال زیر یک شبیه‌سازی از این سلکشن را نشان می‌دهد که در آن ۴۰ درصد از ۱۰ کروموزوم در مسابقات انتخاب می‌شوند. روش انتخاب مسابقات با شبه کد توضیح داده می‌شود:

```
choose k (the tournament size) individuals from the population at random
choose the best individual from the tournament with probability p
choose the second best individual with probability p*(1-p)
choose the third best individual with probability p*((1-p)^2)
and so on
```

مسابقه قطعی، بهترین فرد را (در صورت $p=1$) در هر تورنمنت انتخاب می‌کند. یک انتخاب یک طرفه یعنی زمانی که ($k=1$) است معادل انتخاب تصادفی است.

اگر ۱۰۰ کروموزوم در جمعیتی با ۱۰ درصد **elite selection** وجود داشته باشد، باید ۹۰ بار مسابقات ایجاد کنیم تا ۴۵ کروموزوم را جفت کنیم که ۹۰ فرزند تولید می‌کند. کروموزوم‌های فرزندان و شایسته پس از تلاقی در مجموع به ۱۰۰ می‌رسد که تعداد جمعیت در تمام طول فرآیند الگوریتم ژنتیک است.

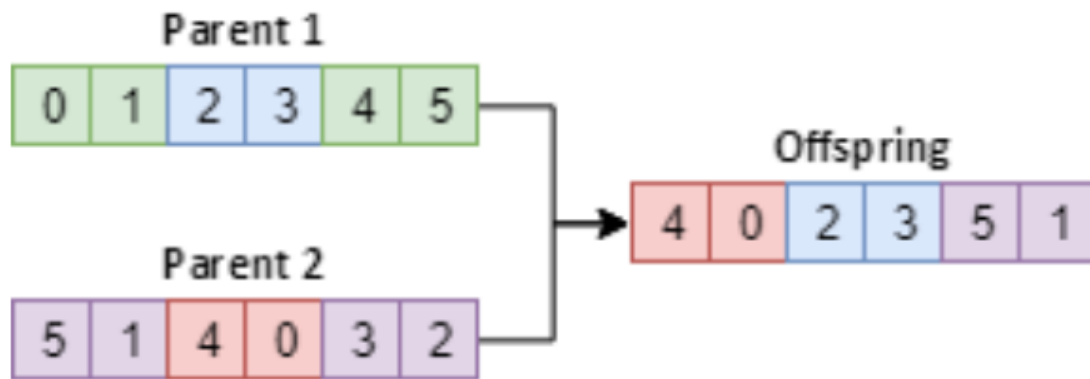
Selection^{۱۵}

Tournament Selection^{۱۶}

جفت گیری^{۱۷} و تولید نسل جدید^{۱۸}

استخر جفت گیری^{۱۹} شامل کروموزوم های انتخابی است که قرار است با یکدیگر کراس اور^{۲۰} انجام دهند. عملیات کراس اور شامل تولید فرزندان بر اساس ژن های والدین مشابه با همتای بیولوژیکی آن است. روش های متعددی برای انجام کراس اور وجود دارد و مانند انتخاب، استفاده از مناسب ترین روش برای داشتن نتیجه بهتر مهم است.

Order crossover^{۲۱} برای مسائل مبتنی بر جایگشت بهتر عمل می کند. این فرآیند بدین صورت است که خوشه ژنی والد اول را به فرزندان می برد و سپس خوشه ژنی والد دوم را به فرزندان می پیچد. برای هر جفت کروموزوم، دو فرزند تولید می شود که نقش والدین برای تولید فرزندان متفاوت معکوس می شود. همه فرزندان جای والدین خود را در جمعیت خواهند گرفت. در شکل زیر نحوه ی اجرا این نوع کراس اور مشخص است:



^{۱۷} Crossover

^{۱۸} Offspring

^{۱۹} Mating pool

^{۲۰} Crossover

^{۲۱} Order Crossover (ox)

جهش

جهش یک عملگر ژنتیکی است که به تنوع در جمعیت کمک می کند. همچنین می تواند از گیر کردن راه حل ها در بهینه محلی^{۲۲} جلوگیری کند. جهش فرآیند تغییر ژن/های یک کروموزوم است. معمولاً میزان جهش پایین و ثابت است. با این حال، الگوریتم ژنتیک تطبیقی به این معنی است که نرخ جهش ممکن است بسته به عملکرد جمعیت متفاوت باشد. که در بخش های قبل نحوه ی تغییر و فرمول تغییر نرخ جهش را کامل بررسی کردیم.

دامنه و محدودیت

هدف این تحقیق ایجاد یک هوش مصنوعی است که می تواند جدول زمانی ایجاد کند. که فقط پردازش داده های ورودی و تولید نتایج معقول انسان را پوشش می دهد. فرآیندی که در ایجاد زمان بندی استفاده می شود، الگوریتم ژنتیک تطبیقی-شایسته گرا^{۳۳} است. نوعی الگوریتم تکاملی که شامل حفظ بهترین مجموعه راه حل ها در نسل های بعدی برای حفظ راه حل های سطح بالا است، اما در عین حال، متغیرهایی مانند تعداد جمعیت، نرخ جهش و فشار انتخاب تغییر می کنند تا از هم گرایی زودرس جلوگیری شود که منجر به خروجی ضعیف خواهد شد.

محاسبه زمان بندی شامل اعتبارسنجی برای هر محدودیت است. محدودیت ها مجموعه قوانینی هستند که پذیرش خروجی را هدایت می کنند. سه نوع محدودیت برای این مسئله وجود خواهد داشت.

۱. **محدودیت های نرم:** مجموعه ای از قوانین که می تواند بدون تأثیر بر اعتبار خروجی شکسته شود.
۲. **محدودیت های متوسط:** مجموعه ای از قوانین که می تواند با تأثیر بر اعتبار خروجی شکسته شود. با این حال، این تنها زمانی می تواند شکسته شود که سناریو از نظر منطقی نامعتبر یا غیرممکن باشد.
۳. **محدودیت های سخت:** مجموعه ای از قوانین که در صورت شکسته شدن، راه حل نامعتبر ایجاد می کند.

^{۲۲} Local Beam

^{۳۳} adaptive-elitist genetic algorithm

خروجی نرم افزار

تضمین نمی شود که نتایج برنامه بهترین راه حل ممکن برای سناریو باشد. کیفیت نتیجه به شدت به جهت گیری اجرا بستگی دارد. به دلیل ماهیت تصادفی آن، نتایج ممکن است از ضعیف تا عالی متفاوت باشد. سیستم تضمین نمی کند که محدودیت های سخت هر راه حل برآورده شود، به ویژه زمانی که سناریوی ارائه شده از نظر منطقی غیرممکن یا به شدت فشرده باشد. این همچنین به این معنی است که وقتی در یک سناریوی سخت، محدودیت های نرم یا متوسط در معرض نقض هستند تا محدودیت های سخت برآورده شود. محدودیت های سخت باید رعایت شوند و هرگز نقض نخواهند شد.

تنظیمات الگوریتم ژنتیک

پیکربندی الگوریتم ژنتیک در برنامه برای همه نوع مسائل یکسان نیست که بتواند هر سناریوی را برآورده کند. این به دلیل محدودیت قدرت محاسباتی است. برای مقابله با محدودیت و همچنین پشتیبانی از مقیاس پذیری، تنظیمات اجرا الگوریتم ژنتیک را می توان با توجه به قابلیت یا اولویت اپراتور تغییر داد. جدول زیر شرح و توصیفی برای هر ویژگی قابل تغییر الگوریتم را نشان می دهد.

| تنظیمات | توضیحات | محدودیت |
|----------------------------------|--|--------------|
| حداقل تعداد جمعیت ^{۲۴} | حداقل و تعداد اولیه کروموزوم ها در یک نسل. پنجاه (۵۰) مقدار پیشنهادی است زیرا برای انواع راه حل ها به اندازه کافی زیاد است. | ۵۰- ۱۰۰۰۰ |
| حداکثر تعداد جمعیت ^{۲۵} | محدودیت در تعداد کروموزوم های یک نسل. محاسبه میزان تحمل دستگاه محاسباتی به میزان رم آن بستگی دارد. به ازای هر هفتاد (۷۰) موجودیت فعال در یک سناریو (موضوع، مربی، اتاق و بخش)، یک مگابایت (۱ مگابایت) حافظه در هر کروموزوم مصرف می کند. بنابراین، اگر از پنجاه (۵۰) موجودیت و حداکثر تعداد جمعیت صد (۱۰۰) استفاده شود، پیک مصرف RAM تقریباً ۷۲ مگابایت به اضافه ۲۵ درصد مقدار برای مدیریت برنامه اصلی خواهد بود. یک کامپیوتر رده پایین باید حداکثر تعداد جمعیت را پایین نگه دارد، می تواند با حداقل پیشنهاد جمعیت پنجاه (۵۰) مطابقت داشته باشد. | ۵۰- ۱۰۰۰۰ |
| حداکثر تعداد نسل ^{۲۶} | محدودیت تعداد نسلی که الگوریتم می تواند تولید کند. با رسیدن به مقدار تنظیم شده، الگوریتم متوقف می شود. هرچه تعداد نسل بیشتر یا سناریو محدودیت های پیچیده تری داشته باشد، بیشتر باید اجرا شود، زیرا شانس بیشتری به هوش مصنوعی می دهد تا مشکلات جزئی را برطرف کند یا حتی پیشرفتی ایجاد کند که می تواند کل مجموعه راه حل ها را تغییر دهد. | ۵۰- ۱۰۰۰۰ |

Minimum Population Count ^{۲۴}

Maximum Population ^{۲۵}

Maximum Generations ^{۲۶}

| | | |
|-----------------|---|---|
| ۱۵۰۰- ۳۰۰۰ | حداکثر تلاش هوش مصنوعی برای قرار دادن یک برنامه زمانی معتبر در کروموزوم. یک کامپیوتر قوی می تواند ارزش بیشتری برای این کار قائل شود، زیرا این بیشتر به عنوان یک تاکتیک brute force عمل می کند که می تواند به افزایش کیفیت راه حل کمک کند. | حداکثر کروموزم جدید ^{۲۷} |
| ۰.۰۰- ۱۰۰.۰۰ | مقدار آستانه برای شروع تغییر نرخ جهش. این مقدار برابر است با تفاوت آخرین میانگین فیتنس را با میانگین فیتنس نسل قبلی آن. اگر تفاوت کمتر یا برابر با مقدار تنظیم شده باشد، نرخ جهش ۵٪ افزایش می یابد و در غیر این صورت تا ۵٪ کاهش می یابد. اگر اپراتور نمی خواهد نرخ جهش تغییر کند، مقدار را می تواند روی صفر تنظیم کرد. توصیه می شود برای جلوگیری از تغییرات گسترده در جمعیت، مقدار این تنظیم را روی پیش فرض ۰.۰۸ نگه دارید. | فعال کننده نرخ جهش ^{۲۸} |
| ۰-۱۰۰٪ | ماشه ای برای پایان دادن به الگوریتم. هنگامی که فیتنس یک کروموزوم به مقدار تنظیم شده رسید، الگوریتم متوقف می شود. پیشنهاد می شود برای این مقدار یک مقدار بهینه قرار گیرد که می تواند از ۹۰ تا ۹۸ درصد برای سطح قابل قبول متغیر باشد. | حداکثر مقدار فیتنس ^{۲۹} |
| ۰-۱۰۰٪ | درصدی از اینکه چه مقدار از کروموزوم های جمعیت فعلی با بهترین عملکرد تا نسل بعدی حفظ می شود. مقدار پیشنهادی برای متغیر ۵٪ است زیرا تعداد کروموزوم های خوب را پایین نگه می دارد و به جلوگیری از همگرایی زودرس کمک می کند. | جمعیت شایسته ^{۳۰} |
| ۰-۱۰۰٪ | غناي مجموعه راه حل جمعیت را اندازه گیری می کند. تناسب هر کروموزوم برای انحراف آماری اندازه گیری و نرمال می شود. اگر یکی از سیگماهای انحراف از مقدار تعیین شده فراتر رود، تعداد جمعیت به میزان درصدی تغییر می کند. در غیر این صورت جمعیت به میزان درصد کمبود آن کاهش می یابد. این به حفظ تنوع بر اساس انحراف تناسب بین کروموزوم ها کمک می کند. مقدار پیشنهادی ۵۵٪ است زیرا از تسلط کامل یک گروه مشترک از کروموزوم ها جلوگیری می کند. | انحراف تحمل ^{۳۱} |

Maximum Creation Attempt ^{۲۷}

Mutation Rate Adjustment Trigger ^{۲۸}

Maximum Fitness ^{۲۹}

Elite Population ^{۳۰}

Deviation Telorance ^{۳۱}

فصل پنج : رابط گرافیکی

برای استفاده راحت تر کاربر نیاز به یک رابط گرافیکی احساس می شد.

ساختار کلی:

پنجره اصلی:

پنجره اصلی که در زمان باز شدن برنامه مشاهده خواهد شد شامل زبانه های زیر است :

۱. اساتید
۲. کلاس ها (اتاق ها)
۳. درس ها
۴. گروه ها
۵. مدیریت سناریو

که به صورت پیشفرض زبانه اساتید فعال است.

زبانه های اساتید، کلاس ها و درس ها شامل سه بخش هستند :

۱. عملیات: در این بخش می توان به ساختمان داده اعضای مربوط به هر کدام از زبانه ها را اضافه کرد. همچنین دکمه ای برای بارگذاری از فایل اکسل (Excel) در نظر گرفته شده است.
۲. جست و جو: این بخش برای دسترسی سریع تر به داده مورد نظر برای تغییر یا حذف آن ایجاد شده است.
۳. نمایش داده ها: در این بخش داده های مربوط به هر زبانه نمایش داده می شود.

نمایش داده ها:

در پنجره اصلی در صورت وجود داده مربوط به هر زبانه مشابه (تصویر ۱-۵) نمایش داده می شود. و شما می توانید برخی از اطلاعات مهم را مشاهده کنید. برای مثال می توانید اسم اساتید را در زبانه اساتید مشاهده کنید. (تصویر ۵-۱)

امکان مرتب کردن هر ستون بر اساس حروف الفبا برای هر ستون که حاوی اطلاعات است، با کلیک بر روی سرستون ممکن است.



تصویر ۵-۱

ستون عملیات

با کلیک بروی آیکون ادیت (اولین آیکون از سمت راست) می توان اطلاعات آن ردیف را تغییر داد. همچنین با کلیک بروی آیکون حذف (دومین آیکون) می توان آن را حذف کرد.

ستون در دسترس

این ستون برای فعال کردن یا غیر فعال کردن کلاس ها، استاد ها و گروه های مورد نظر است. به طوری که با غیر فعال کردن استادی، آن استاد در عملیات زمانبندی در نظر گرفته نمی شود. (تصویر ۵-۱)

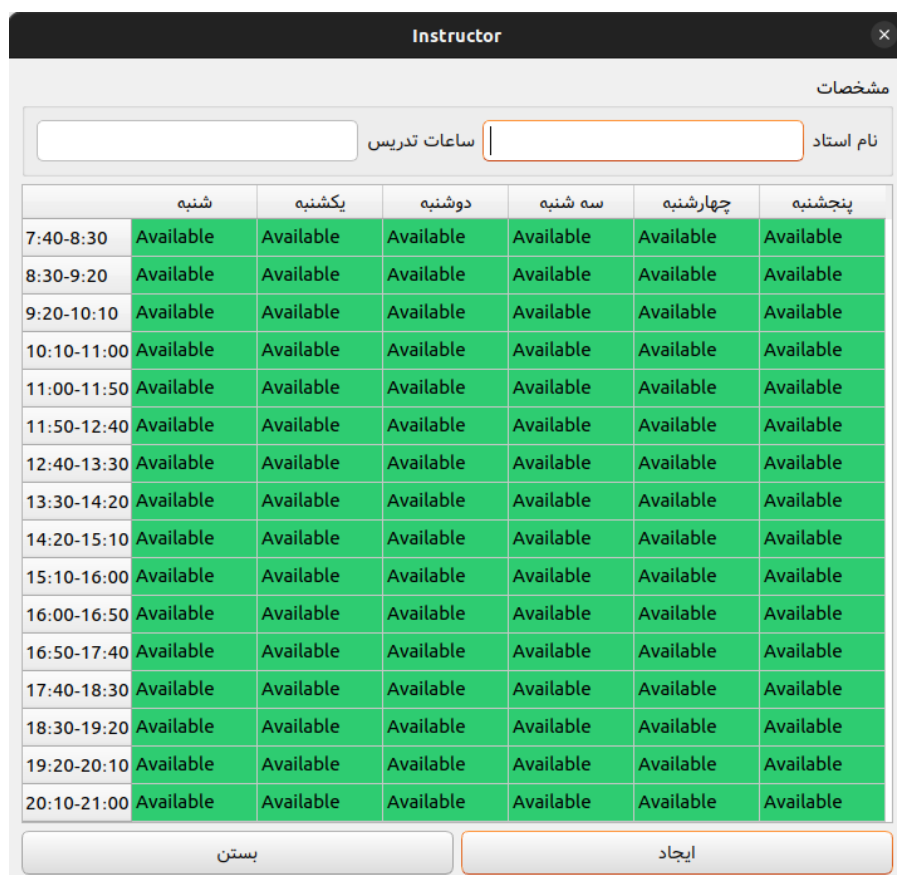
جدول زمانی

این جدول (تصویر ۵-۲)، به صورت پیشفرض تماماً در دسترس "Available" است. و شما می توانید با کلیک کردن بروی هر خانه جدول وضعیت آن خانه را تغییر دهید. همچنین با کلیک کردن بر روی سر ستون ها (شنبه، یک شنبه، دوشنبه و ...)، وضعیت آن روز را در تمامی ساعات غیر قابل دسترس "Unavailable" کنید. و با دوبار کلیک کردن به وضعیت در دسترس تغییر دهید. به صورت مشابه برای سر ردیف ها نیز قابل استفاده است.

اساتید

در این قسمت می توان داده های مربوط به اساتید را مشاهده، تغییر، اضافه و یا حذف نمود. (تصویر ۵-۱).

برای اضافه کردن استاد جدید می توان بر روی دکمه **اضافه کردن استاد جدید** کلیک کرد. زیر پنجره باز شده مطابق تصویر ۵-۲ می باشد.



مشخصات

نام استاد: ساعات تدریس:

| | شنبه | یکشنبه | دوشنبه | سه شنبه | چهارشنبه | پنجشنبه |
|-------------|-----------|-----------|-----------|-----------|-----------|-----------|
| 7:40-8:30 | Available | Available | Available | Available | Available | Available |
| 8:30-9:20 | Available | Available | Available | Available | Available | Available |
| 9:20-10:10 | Available | Available | Available | Available | Available | Available |
| 10:10-11:00 | Available | Available | Available | Available | Available | Available |
| 11:00-11:50 | Available | Available | Available | Available | Available | Available |
| 11:50-12:40 | Available | Available | Available | Available | Available | Available |
| 12:40-13:30 | Available | Available | Available | Available | Available | Available |
| 13:30-14:20 | Available | Available | Available | Available | Available | Available |
| 14:20-15:10 | Available | Available | Available | Available | Available | Available |
| 15:10-16:00 | Available | Available | Available | Available | Available | Available |
| 16:00-16:50 | Available | Available | Available | Available | Available | Available |
| 16:50-17:40 | Available | Available | Available | Available | Available | Available |
| 17:40-18:30 | Available | Available | Available | Available | Available | Available |
| 18:30-19:20 | Available | Available | Available | Available | Available | Available |
| 19:20-20:10 | Available | Available | Available | Available | Available | Available |
| 20:10-21:00 | Available | Available | Available | Available | Available | Available |

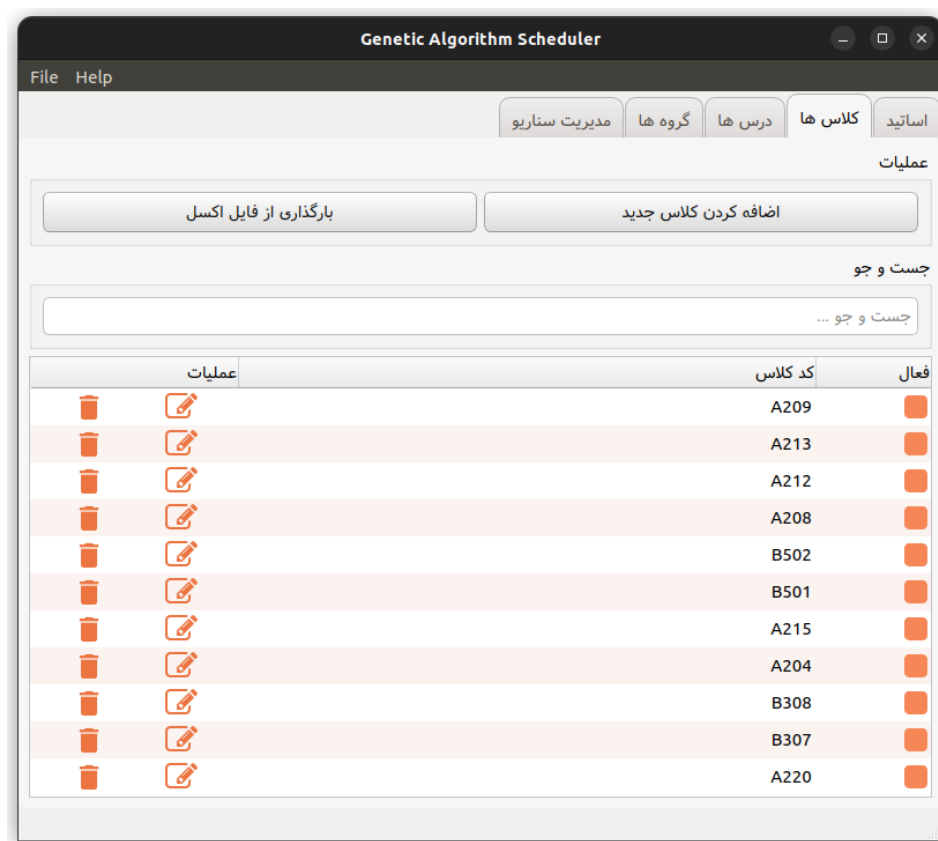
ایجاد بستن

تصویر ۵-۲

۱. نام استاد: نام و نام خانوادگی استاد مورد نظر.
۲. ساعات تدریس: مجموع ساعات تدریس استاد در هفته.
۳. جدول زمانی: در این قسمت روز ها و ساعاتی که استاد امکان تدریس دارد وارد می شد.
۴. دکمه ایجاد: با کلیک بر روی این دکمه اطلاعات در پایگاه داده ثبت و زیرپنجره بسته می شود.
۵. دکمه بستن: بدون تغییر در پایگاه داده زیر پنجره را می بندد.

کلاس ها

در این قسمت می توان داده های مربوط به کلاس ها را مشاهده، تغییر، اضافه و یا حذف نمود. (تصویر ۵-۳)



تصویر ۵-۳

برای اضافه کردن کلاس بر روی دکمه **اضافه کردن کلاس جدید** کلیک کنید. زیر پنجره باز شده مطابق تصویر ۵-۴ می باشد.

مطابق تصویر ۵-۴ کد کلاس را در قسمت مشخصات و نوع کلاس را وارد می شود. و با استفاده از جدول زمانی زمان های در

دسترس بودن کلاس را می توان مشخص کرد.

Room
×

نوع کلاس

☐ نظری
 ☒ عملی

مشخصات

کد کلاس

| | شنبه | یکشنبه | دوشنبه | سه شنبه | چهارشنبه | پنجشنبه |
|-------------|-----------|-----------|-----------|-----------|-----------|-----------|
| 7:40-8:30 | Available | Available | Available | Available | Available | Available |
| 8:30-9:20 | Available | Available | Available | Available | Available | Available |
| 9:20-10:10 | Available | Available | Available | Available | Available | Available |
| 10:10-11:00 | Available | Available | Available | Available | Available | Available |
| 11:00-11:50 | Available | Available | Available | Available | Available | Available |
| 11:50-12:40 | Available | Available | Available | Available | Available | Available |
| 12:40-13:30 | Available | Available | Available | Available | Available | Available |
| 13:30-14:20 | Available | Available | Available | Available | Available | Available |
| 14:20-15:10 | Available | Available | Available | Available | Available | Available |
| 15:10-16:00 | Available | Available | Available | Available | Available | Available |
| 16:00-16:50 | Available | Available | Available | Available | Available | Available |
| 16:50-17:40 | Available | Available | Available | Available | Available | Available |
| 17:40-18:30 | Available | Available | Available | Available | Available | Available |
| 18:30-19:20 | Available | Available | Available | Available | Available | Available |
| 19:20-20:10 | Available | Available | Available | Available | Available | Available |
| 20:10-21:00 | Available | Available | Available | Available | Available | Available |

بستن

ایجاد

تصویر ۴-۵

درس ها

در این قسمت می توان داده های مربوط به درس ها را مشاهده، تغییر، اضافه و یا حذف نمود. تصویر ۵-۵

برای اضافه کردن درس بر روی دکمه **اضافه کردن درس جدید** کلیک کنید. زیر پنجره باز شده مطابق تصویر ۶-۵ می باشد.

در پنجره جدید نام درس کد درس و مجموع ساعات درسی را وارد کنید (یک درس نظری ۳ واحدی، ۳ ساعت آموزشی، و یک درس ۱ واحدی عملی، ۲ ساعت آموزشی را شامل می شود).

در صورت نیاز می توانید برای هر درس توضیحاتی نیز بنویسید که این توضیحات در الگوریتم استفاده نمی شود و صرفاً برای رفع نیاز کاربر نگاه داری می شوند.

در ادامه با فعال کردن اساتید مورد نظر برای آن درس می توانید فرایند ایجاد درس جدید را تکمیل کنید. در صورتی می خواهید از یک درس چند کد ایجاد کنید می توانید از دکمه **ایجاد** استفاده کرده و مقادیر را نگاه دارید و تنها کد کلاس تغییر دهید.

Genetic Algorithm Scheduler

File Help

اساتید کلاس ها درس ها گروه ها مدیریت سناریو

عملیات

بارگذاری از فایل اکسل اضافه کردن درس جدید

جست و جو

جست و جو ...

| کد درس | نام درس | نوع درس | اساتید | عملیات |
|--------|------------------------------|---------|--------|--------|
| AE-4 | زبان تخصصی | LEC | اساتید | |
| DPL-1 | طراحی زبان‌های برنامه‌سازی | LEC | اساتید | |
| DBL-5 | آز پایگاه داده | LAB | اساتید | |
| DBL-4 | آز پایگاه داده | LAB | اساتید | |
| OSL-4 | آز سیستم عامل | LEC | اساتید | |
| OSL-3 | آز سیستم عامل | LAB | اساتید | |
| MP-1 | مبانی کامپیوتر و برنامه‌سازی | LEC | اساتید | |
| AP-6 | یک‌پارچه‌سازی کاربردها | LEC | اساتید | |
| EPK-1 | یک‌پارچه‌سازی کاربردها | LEC | اساتید | |
| DOS-3 | طراحی شی‌گرای سیستم‌ها | LEC | اساتید | |
| AP-1 | برنامه‌سازی پیشرفته | LEC | اساتید | |

تصویر ۵-۵

Subject

نام درس ساعات درسی

کد درس توضیحات

نوع درس

هر دو ☐ نظری ☒ عملی ☐

جست و جو

جست و جو ...

| فعال | استاد |
|--------------------------|--------|
| <input type="checkbox"/> | اساتید |
| <input type="checkbox"/> | اساتید |
| <input type="checkbox"/> | اساتید |
| <input type="checkbox"/> | اساتید |
| <input type="checkbox"/> | اساتید |
| <input type="checkbox"/> | اساتید |
| <input type="checkbox"/> | اساتید |
| <input type="checkbox"/> | اساتید |

بستن ایجاد ایجاد و بستن

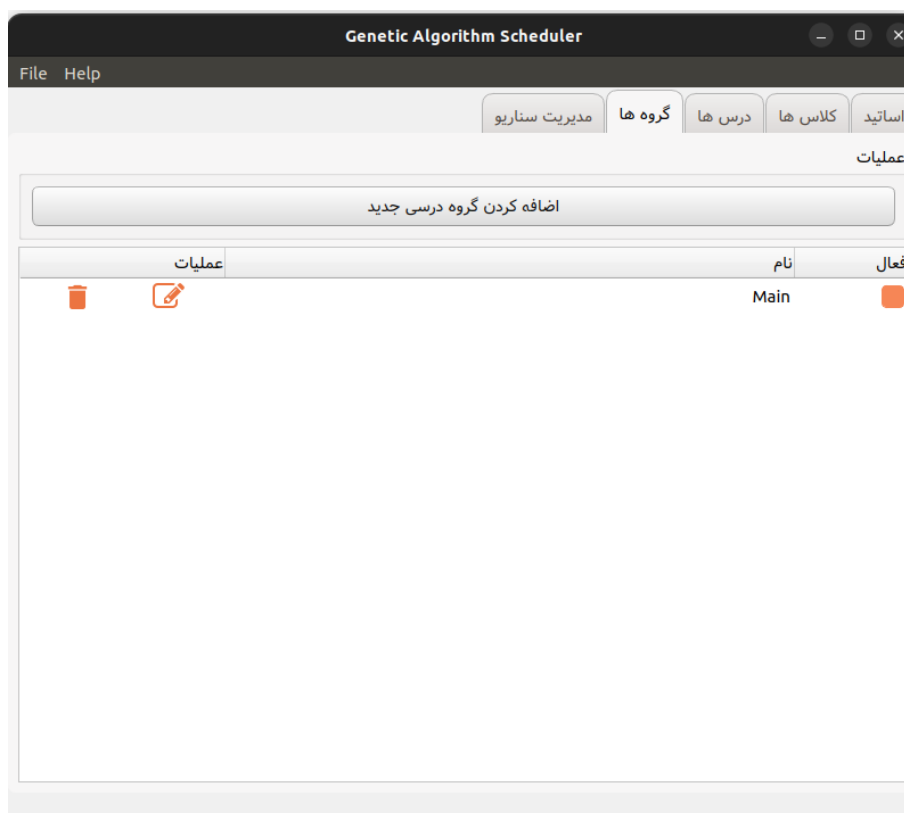
تصویر ۵-۶

گروه ها

در این قسمت می توان داده های مربوط به گروه ها را مشاهده، تغییر، اضافه و یا حذف نمود. تصویر ۵-۷

برای اضافه کردن گروه بر روی دکمه **اضافه کردن گروه درسی جدید** کلیک کنید. زیر پنجره باز شده مطابق تصویر ۵-۸ می باشد.

در پنجره باز شده نام گروه را وارد کرده و درس هایی که می خواهید از فهرست انتخاب کنید، می توانید از قسمت جست و جو کمک بگیرید، در ادامه وارد کرد زمان هایی که می خواهید از آن گروه کلاس ایجاد شود را در جدول زمانبندی وارد کنید.



تصویر ۵-۷

Section

نام گروه

جست و جو در واحد ها

جست و جو ...

| فعال | کد درس | نام درس |
|--------------------------|--------|----------------------------|
| <input type="checkbox"/> | AE-4 | زبان تخصصی |
| <input type="checkbox"/> | DPL-1 | طراحی زبان‌های برنامه‌سازی |
| <input type="checkbox"/> | DBL-5 | آز پایگاه داده |
| <input type="checkbox"/> | DBL-4 | آز پایگاه داده |
| <input type="checkbox"/> | OSL-4 | آز سیستم عامل |
| <input type="checkbox"/> | OSL-3 | آز سیستم عامل |

| | شنبه | یکشنبه | دوشنبه | سه شنبه | چهارشنبه | پنجشنبه |
|-------------|-----------|-----------|-----------|-----------|-----------|-----------|
| 7:40-8:30 | Available | Available | Available | Available | Available | Available |
| 8:30-9:20 | Available | Available | Available | Available | Available | Available |
| 9:20-10:10 | Available | Available | Available | Available | Available | Available |
| 10:10-11:00 | Available | Available | Available | Available | Available | Available |
| 11:00-11:50 | Available | Available | Available | Available | Available | Available |
| 11:50-12:40 | Available | Available | Available | Available | Available | Available |
| 12:40-13:30 | Available | Available | Available | Available | Available | Available |
| 13:30-14:20 | Available | Available | Available | Available | Available | Available |
| 14:20-15:10 | Available | Available | Available | Available | Available | Available |
| 15:10-16:00 | Available | Available | Available | Available | Available | Available |
| 16:00-16:50 | Available | Available | Available | Available | Available | Available |
| 16:50-17:40 | Available | Available | Available | Available | Available | Available |
| 17:40-18:30 | Available | Available | Available | Available | Available | Available |
| 18:30-19:20 | Available | Available | Available | Available | Available | Available |
| 19:20-20:10 | Available | Available | Available | Available | Available | Available |
| 20:10-21:00 | Available | Available | Available | Available | Available | Available |

بستن

ایجاد

تصویر ۵-۸

مدیریت سناریو

ایجاد برنامه جدید، تغییر تنظیمات و دریافت فایل اکسل (Excel) خروجی در این زبانه امکان پذیر است. که در چهار بخش عملیات، تنظیمات عملیات، تنظیمات الگوریتم ژنتیک و ماتریس ارزیابی دسته بندی شده است. (تصویر ۵-۹)

تصویر ۵-۹

ایجاد

این پنجره ای است که هنگام انجام الگوریتم دیده می شود، در این پنجره می توانید اطلاعات وضعیت سیستم مراحل الگوریتم ژنتیک را مشاهده کنید. همچنین بعد از ایجاد نسل اولیه بهترین هر نسل را می توان مشاهده، و یا برای بالاتر بردن سرعت عملیات آن را غیر فعال کرد. (تصویر ۵-۱۰)



تصویر ۱۰-۵

ساختار کد

ابزار های مورد استفاده

۱. برای طراحی رابط گرافیکی از ابزار Qt Disingner استفاده شد است.
۲. برای ساخت و طراحی توابع مربوط به رابط گرافیکی از کتابخانه PyQt استفاده شده است.
۳. برای ایجاد icon های داخل برنامه از کتابخانه qtawesome استفاده شده است.

طراحی GUI

طراحی شامل یک پنجره اصلی:

۱. Main

و چند زیر پنجره است :

۲. Instructor

۳. Subject

۴. Room

Section ۵. Generate ۶.

زیر پنجره ها ۲ تا ۵ برای وارد کردن اطلاعات مربوط به هر کدوم از بخش ها مورد استفاده قرار می گیرد. زیر پنجره ۶ در هنگام اجرا الگوریتم نمایش داده می شود، که پیشتر مشاهده کردیم.

اجزای GUI

فایل های UI

فایل های طراحی شده با فرمت ui در پوشه "qt_ui" در دسترس هستند.

فایل های تبدیل شده UI به py

فایل تبدیل شده به python هر پنجره در پوشه "py_ui"، و توابع مربوط به هر کدام در پوشه "containers" قرار دارد.

فایل های موجود در پوشه "components"

۱. TableModel.py : کلاس استاندارد برای ساخت جداول.

۲. Timetable.py : کلاس Timetable برای ایجاد جدول زمانی.

۳. PreviewScheduleParser.py : کلاس مربوط به جدول پیش نمایش که در پنجره Generate استفاده می شود.

فایل های Style

برای ایجاد تغییر در Style پنجره ها از QSS استفاده شده است، که فایل مربوط به آن در پوشه "assets/style" قرار دارد.

Py_ui

فایل Main.py کلاس Ui_MainWindow و سایر فایل ها کلاس Ui_Dialog دارند. این کلاس ها از کلاس های استاندارد PyQt هستند که برای ساخت اجزای گرافیکی پنجره ها GUI ما مورد استفاده قرار می گیرند. که با فراخوانی تابع setupUi می توان این اجزا را ساخت، البته در این جا ساخت این فایل ها به خود PyQt واگذار شده، در واقع ما با استفاده از Qt Designer و ساخت فایل ui مورد نظر تحت محیط گرافیکی و تبدیل این فایل با استفاده از دستور

```
python -m PyQt5.uic.pyuic -x [FILENAME].ui -o [FILENAME].py
```

آن را به فایل python تبدیل کردیم. البته این فایل ساخته شده صرفا ساختار گرافیکی بوده و عملکردی ندارد.

Containers

توابع مربوط به GUI در این پوشه قرار دارد که عملکرد مربوط به دکمه ها و ... را شامل می شود.

Instructor

توابع مربوط به زبانه اساتید و زیرپنجره مربوطه را در ادامه بررسی می کنیم.

در ابتدا باید کتابخانه ها و توابع فریمورک qt را به پروژه وارد کنیم. (خط ۱)، سپس پایگاه داده و اجزای مرتبط را به پروژه متصل کرده. (خط ۲)، وارد کردن فایل ساختار زیر پنجره اساتید به عنوان Parent، و کتابخانه های مهم json، os، ... را به برنامه اضافه می کنیم. (خط ۴ و ۵)، و در آخر کتابخانه مربوط برای ساخت آیکون ها.

```
from PyQt5 import QtWidgets, QtGui, QtCore
from components import Database as db, Timetable
from py_ui import Instructor as Parent
import json
import os
import qtawesome as qta
```

کلاس های تعریف شده در پروژه که خود شامل متد هایی می شوند :

کلاس Instructor

گرفتن ورودی و ساخت فیلد در دیتابیس و بررسی خطا ها برای اساتید.

```
class Instructor:
    def __init__(self, id):
        self.id = id
        self.dialog = dialog = QtWidgets.QDialog()
        # From the qt_ui generated UI
        self.parent = parent = Parent.Ui_Dialog()
        parent.setupUi(dialog)
        if id:
            self.fillForm()
        else:
            # Create a new instance of timetable
            self.table = Timetable.Timetable(parent.tableSchedule)
        parent.btnFinish.clicked.connect(self.finish)
        parent.btnCancel.clicked.connect(self.dialog.close)
        dialog.exec_()
```

هنگامی که بر روی گزینه ادیت کلیک می شود تابع زیر اطاعات آن استاد را از پایگاه داده دریافت و در زیرپنجره باز شده وارد می کند.

```
def fillForm(self):
    conn = db.getConnection()
    cursor = conn.cursor()
    cursor.execute('SELECT name, hours, schedule FROM instructors WHERE id = ?',
[self.id])
    result = cursor.fetchone()
    conn.close()
    self.parent.lineEditName.setText(str(result[0]))
    self.parent.lineEditHours.setText(str(result[1]))
    # Generate timetable from custom schedule
    self.table = Timetable.Timetable(self.parent.tableSchedule,
json.loads(result[2]))
```

ثبت اطلاعات وارد شده در پایگاه داده در صورت صحیح بودن.

```
def finish(self):
    # Verification of input
    if not self.parent.lineEditName.text():
        self.error('لطفا نام استاد را وارد کنید')
        return False
    name = self.parent.lineEditName.text()
    try:
        hours = int(self.parent.lineEditHours.text())
        if hours <= 0 or hours > 100 or hours % .5 != 0:
            self.error('لطفا میزان ساعت فعالیت استاد در هفته را در بازه مجاز وارد کنید')
            return False
    except:
        self.error('لطفا میزان ساعت فعالیت استاد در هفته را مشخص کنید')
        return False
    data = [name, hours, json.dumps(self.table.getData()), self.id]
    if not self.id:
        data.pop()
    self.insertInstructor(data)
    self.dialog.close()
```


تابع زیر مربوط به ساخت پیغام های خطا در GUI است که در تابع بالا فراخوانی شده اند.

```
def error(self, message):
    confirm = QtWidgets.QMessageBox()
    confirm.setIcon(QtWidgets.QMessageBox.Warning)
    confirm.setText(message)
    confirm.setWindowTitle('خطا')
    confirm.setStandardButtons(QtWidgets.QMessageBox.Ok)
    confirm.exec_()
```

اضافه کردن اساتید به پلیگاه داده ها، در صورت وجود آن استاد (در صورت ویرایش شدن اطلاعات)، اطلاعات آپدیت شده و در غیر این صورت اطلاعات به پایگاه داده وارد می شوند.

```
@staticmethod
def insertInstructor(data):
    conn = db.getConnection()
    cursor = conn.cursor()
    if len(data) > 3:
        cursor.execute('UPDATE instructors SET name = ?, hours = ?, schedule
        = ? WHERE id = ?', data)
    else:
        cursor.execute('INSERT INTO instructors (name, hours, schedule)
        VALUES (?, ?, ?)', data)
    conn.commit()
    conn.close()
    return True
```

کلاس *SortFilterProxyModel*

برای ایجاد جست و جو نیاز به استفاده از **ProxyModel** است. این کلاس بعنوان واسط میان اطلاعات دریافتی از پایگاه داده و اطلاعات ورودی کلاس **Tree** عمل کرده و با جست جو در فیلدها، آن هایی را که تشابه دارند را جدا می کند و برای نمایش در اختیار کلاس **Tree** قرار می دهد.

```
class SortFilterProxyModel(QtCore.QSortFilterProxyModel):
    def __init__(self, *args, **kwargs):
        QtCore.QSortFilterProxyModel.__init__(self, *args, **kwargs)
        self.filters = {}
```

مرتب سازی ستون در فیلتر شده.

```
def setFilterByColumn(self, regex, column):
    self.filters[column] = regex
    self.invalidateFilter()
```

فیلتر کردن فیلد ها بر اساس ریجکس دریافتی.

```
def filterAcceptsRow(self, source_row, source_parent):
    for key, regex in self.filters.items():
        ix = self.sourceModel().index(source_row, key, source_parent)
        if ix.isValid():
            text = self.sourceModel().data(ix)
            if not regex in text:
                return False
    return True
```

کلاس Tree

این کلاس وظیفه نمایش لیست اساتید در زبانه اساتید را بر عهده دارد. این کلاس با فراخوانی مدل `tree` از `QtGui.QStandardItemModel()` اقدام به ساخت ساختار این جدول شامل سرآیندها می نماید. و پس از دریافت اطلاعات از پایگاه داده اطلاعات را نمایش می دهد.

```
class Tree:
    def __init__(self, tree):
        self.tree = tree
        self.model = model = QtGui.QStandardItemModel()
        model.setHorizontalHeaderLabels(['id', 'درسترس', 'نام', 'ساعات حضور', 'عملیات'])
        self.proxyModel = proxyModel = SortFilterProxyModel(
            tree, recursiveFilteringEnabled=True
        )
        self.proxyModel.setSourceModel(self.model)
        tree.setModel(proxyModel)
        tree.setColumnHidden(0, True)
        tree.header().setDefaultAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignVCenter)
        model.itemChanged.connect(lambda item: self.toggleAvailability(item))
        self.display()
```

تابع زیر برای تغییر وضعیت در دسترس بودن اساتید استفاده می شود.

```
def toggleAvailability(self, item):  
    # Get ID of toggled instructor  
    id = self.model.data(self.model.index(item.row(), 0))  
    newValue = 1 if item.checkState() == 2 else 0  
    conn = db.getConnection()  
    cursor = conn.cursor()  
    cursor.execute('UPDATE instructors SET active = ? WHERE id = ?', [newValue, id])  
    conn.commit()  
    conn.close()
```

تابع زیر برای نمایش اطلاعات در جدول Tree به کار می رود، و ردیف های مربوط به هر استاد شامل گزینه های ستون عملیات را ایجاد می کند. این کار با ساخت یک کلید و تغییر نوشته آن به وسیله کتابخانه qtawesome انجام می پذیرد.

```

def display(self):
    # Clear model
    btnEditIcon = qta.icon('fa.edit', color='#EC7440',color_active='#E25417')
    deletEditIcon = qta.icon('mdi.delete', color='#EC7440',color_active='#E25417')
    self.model.removeRows(0, self.model.rowCount())
    conn = db.getConnection()
    cursor = conn.cursor()
    cursor.execute('SELECT id, active, hours, name FROM instructors')
    result = cursor.fetchall()
    conn.close()
    for instr in result:
        # ID Item
        id = QtGui.QStandardItem(str(instr[0]))
        id.setEditable(False)
        # Availability Item
        availability = QtGui.QStandardItem()
        availability.setCheckable(True)
        availability.setCheckState(2 if instr[1] == 1 else 0)
        availability.setEditable(False)
        # Hours Item
        hours = QtGui.QStandardItem()
        hours.setData(instr[2], QtCore.Qt.DisplayRole)
        hours.setEditable(False)
        # Name Item
        name = QtGui.QStandardItem(instr[3])
        name.setEditable(False)
        # Edit Item / Container for operation buttons
        edit = QtGui.QStandardItem()
        edit.setEditable(False)
        # Append items to model
        self.model.appendRow([id, availability, name, hours, edit])
        # Create a widget group for edit and delete buttons
        frameEdit = QtWidgets.QFrame()
        # Edit buttons
        btnEdit = QtWidgets.QPushButton(btnEditIcon, '')
        btnEdit.setObjectName("btnEdit")
        btnEdit.setFlat(True)
        btnEdit.setIconSize(QtCore.QSize(32, 32))
        btnEdit.setFixedSize(QtCore.QSize(50, 32))
        btnEdit.clicked.connect(lambda state, id=instr[0]: self.edit(id))
        # Delete buttons
        btnDelete = QtWidgets.QPushButton(deletEditIcon, '')
        btnDelete.setObjectName("btnDelete")
        btnDelete.setFlat(True)
        btnDelete.setIconSize(QtCore.QSize(32, 32))
        btnDelete.setFixedSize(QtCore.QSize(50, 32))
        btnDelete.clicked.connect(lambda state, id=instr[0]: self.delete(id))

        frameLayout = QtWidgets.QHBoxLayout(frameEdit)
        frameLayout.setContentsMargins(0, 0, 0, 0)
        frameLayout.addWidget(btnEdit)
        frameLayout.addWidget(btnDelete)
        frameLayout.setObjectName("test")
        # Append the widget group to edit item
        self.tree.setIndexWidget(self.proxyModel.mapFromSource(edit.index()), frameEdit)

    self.tree.setSortingEnabled(True)
    self.tree.setColumnWidth(2, 400)
    self.tree.setAlternatingRowColors(True)

```

هنگامی که در فیلد مربوط به جست و جو زبانه اساتید تغییری ایجاد می شد تابع زیر اقدام به دریافت آن متن کرده و به ترتیب تابع مربوط به جست و جو و تابع نمایش اطلاعات را مجددا فراخوانی می کند.

```
def onSearchTextChanged(self, text):
    self.proxyModel.setFilterByColumn(text,2)
    self.display()
```

تابع زیر به هنگام کلیک بر روی گزینه ویرایش کلاس Instructor فراخوانی می کند. و پس از اتمام فراید اطلاعات جدول را مجددا نمایش می دهد.

```
def edit(self, id):
    Instructor(id)
    self.display()
```

این تابع به هنگام کلیک بر روی گزینه حذف اقدام به نمایش پیغام بروی صفحه کرده و بعد از گرفتن تأییدیه کاربر اقدام به حذف آن داده از پایگاه داده می کند و مجدد جدول را بارگذاری می کند.

```
def delete(self, id):
    # Show confirm model
    confirm = QtWidgets.QMessageBox()
    confirm.setIcon(QtWidgets.QMessageBox.Warning)
    confirm.setText('Are you sure you want to delete this entry?')
    confirm.setWindowTitle('Confirm Delete')
    confirm.setStandardButtons(QtWidgets.QMessageBox.Yes | QtWidgets.QMessageBox.No)
    result = confirm.exec_()
    # 16384 == Confirm
    if result == 16384:
        conn = db.getConnection()
        cursor = conn.cursor()
        cursor.execute('DELETE FROM instructors WHERE id = ?', [id])
        conn.commit()
        conn.close()
        self.display()
```

Subject

توابع مربوط به زیر پنجره و زبانه درس ها :

از توضیح کلاس ها و توابع مشابه خوداری شده است.

کلاس Subject

کلاس اصلی درس ها که در ادامه به توضیح بخش های آن می پردازیم.

تابع **init** این کلاس ابتدا به ایجاد صفحه و اتصال دکمه ها و فیلد جست و جو به توابع مربوط به خود.

```
class Subject:
    def __init__(self, id):
        self.id = id
        # New instance of dialog
        self.dialog = dialog = QtWidgets.QDialog()
        # Initialize custom dialog
        self.parent = parent = Parent.Ui_Dialog()
        # Add parent to custom dialog
        parent.setupUi(dialog)
        parent.radioLec.setChecked(True)
        #parent.radioYes.setChecked(True)
        if id:
            self.fillForm()
        self.setupInstructors()
        parent.btnFinish.clicked.connect(self.finish)
        parent.btnCancel.clicked.connect(self.dialog.close)
        parent.btnSave.clicked.connect(self.save)
        parent.txtSelectIns.textChanged.connect(lambda value: self.onSearchTextChanged(value))

        dialog.exec_()
```

به صورت مشابه وظیفه تابع زیر دریافت و وارد کردن اطلاعات در زیر پنجره ایجاد شده به هنگام ویرایش درس مورد نظر است.

```

def fillForm(self):
    conn = db.getConnection()
    cursor = conn.cursor()
    cursor.execute('SELECT name, hours, code, description, type FROM subjects
WHERE id = ?', [self.id])
    result = cursor.fetchone()
    conn.close()
    self.parent.lineEditName.setText(str(result[0]))
    self.parent.lineEditHours.setText(str(result[1]))
    self.parent.lineEditCode.setText(str(result[2]))
    self.parent.lineEditDescription.setText(str(result[3]))
    if result[4] == 'lec':
        self.parent.radioLec.setChecked(True)
    elif result[4] == 'lab':
        self.parent.radioLab.setChecked(True)
    else:
        self.parent.radioAny.setChecked(True)

```

تابع زیر اقدام ساخت لیست اساتید در زیر پنجره ایجاد کلاس می کند، این کار توسط **Tree model** صورت می گیرد، همچنین برای ایجاد قابلیت جست و جو از **Proxy Model** استفاده شده است.

```

def setupInstructors(self):
    self.tree = tree = self.parent.treeSchedule
    self.model = model = QtGui.QStandardItemModel()
    model.setHorizontalHeaderLabels(['ID', 'فعال', 'استاد'])
    self.proxyModel = proxyModel = SortFilterProxyModel(
        tree, recursiveFilteringEnabled=True
    )
    self.proxyModel.setSourceModel(model)
    tree.setModel(proxyModel)
    tree.setColumnHidden(0, True)
    tree.header().setDefaultAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignVCenter)
    conn = db.getConnection()
    cursor = conn.cursor()
    cursor.execute('SELECT id, name FROM instructors WHERE active = 1')
    instructors = cursor.fetchall()
    subjectAssignments = []
    if self.id:
        cursor.execute('SELECT instructors FROM subjects WHERE id = ?', [self.id])
        subjectAssignments = list(map(lambda id: int(id),
            json.loads(cursor.fetchone()[0])))
    conn.close()
    for entry in instructors:
        id = QtGui.QStandardItem(str(entry[0]))
        id.setEditable(False)
        availability = QtGui.QStandardItem()
        availability.setCheckable(True)
        availability.setCheckState(2 if entry[0] in subjectAssignments else 0)
        availability.setEditable(False)
        name = QtGui.QStandardItem(str(entry[1]))
        name.setEditable(False)
        model.appendRow([id, availability, name])
    self.tree.setSortingEnabled(True)
    self.tree.setAlternatingRowColors(True)

```

این تابع به هنگام کلیک بر روی دکمه ایجاد و بستن ابتدا اطلاعات را ذخیره کرده و زیرپنجره را می بندد.

```

def finish(self):
    if self.save():
        self.dialog.close()

```


این تابع اقدام به دریافت اطلاعات از زیرپنجره کرده و در صورت صحیح بودن آن، اطلاعات را در پیگاه داده ذخیره می کند. در غیر این صورت پیغام خطا را به کاربر نشان می دهد.

```
def save(self):
    if not self.parent.lineEditName.text():
        self.error('لطفا نام درس را وارد کنید')
        return False
    if not self.parent.lineEditCode.text():
        self.error('لطفا کد درس را وارد کنید')
        return False
    if not self.parent.lineEditHours.text() or float(self.parent.lineEditHours.text()) < 0
or float(
        self.parent.lineEditHours.text()) > 12 or not (
        float(self.parent.lineEditHours.text()) / .5).is_integer():
        return False
    instructors = []
    for row in range(0, self.model.rowCount()):
        if self.model.item(row, 1).checkState() == 0:
            continue
        instructors.append(self.model.item(row, 0).text())
    name = self.parent.lineEditName.text()
    code = self.parent.lineEditCode.text()
    hours = self.parent.lineEditHours.text()
    description = self.parent.lineEditDescription.text()
    if self.parent.radioLec.isChecked():
        type = 'lec'
    elif self.parent.radioLab.isChecked():
        type = 'lab'
    else:
        type = 'any'
    data = [name, hours, code, description, json.dumps(instructors), type, self.id]
    if not self.id:
        data.pop()
    self.insertSubject(data)
    self.parent.lineEditCode.clear()
    return True
```

Room

** کلاس مشابه کلاس های اساتید.

Section

** کلاس مشابه کلاس های درس ها.