



دانشگاه صنعتی نوشیروانی بابل

برق و کامپیوتر

پروژه کارشناسی رشته مهندسی کامپیوتر گرایش نرم­افزار

اتوماسیون فرآیند انتخاب واحد بوسیله ارائه پیشنهاد برنامه­های ترمی خودکار

نگارش

**طه رستمی**

استاد راهنما

دکتر سيد محمود سخائي

استاد مشاور

آبان 1397

فهرست نوشتار

عنوان صفحه

**فصل اول: مقدمه**

[1-1 موضوع پروژه 2](#_Toc523675084)

[1-2 اهمیت و کاربرد ها 2](#_Toc523675085)

[1-3تعریف صورت مسئله بصورت دقیق­تر 2](#_Toc523675086)

[1-4 اهداف پروژه 3](#_Toc523675087)

[1-5 گام­های انجام پروژه 3](#_Toc523675088)

[1-6 خلاصه 5](#_Toc523675089)

**فصل دوم: طراحی اولیه و آشنایی با مفاهیم پایه**

[2-1 مقدمه 7](#_Toc523675090)

[2-2 فشرده اول از تحلیل و طراحی 7](#_Toc523675091)

[2-3 فشرده دوم از تحلیل و طراحی 9](#_Toc523675092)

[2-3-1 گام کاستن 1 9](#_Toc523675093)

[2-3-2 گام کاستن 2 10](#_Toc523675094)

[2-4 سایر مولفه­ها و سازماندهی بخش­های مربوطه 13](#_Toc523675095)

[2-5 خلاصه 14](#_Toc523675096)

**فصل سوم: ساختارها، الگوریتم­ها و روش­های اصلی حل مسئله**

[3-1 مقدمه 16](#_Toc523675097)

[3-2 ساختار توصیف کننده برنامه درسی مصوب وزارت علوم 16](#_Toc523675098)

[3-2-1 نقص­ها و چالش­ها 20](#_Toc523675099)

[3-3 الگوریتم گام کاستن 1 22](#_Toc523675100)

[3-4 الگوریتم مرحله 1 از گام کاستن 2 23](#_Toc523675101)

[3-5 ساختارها و الگوریتم­های اصلی مرحله2 از گام کاستن2 24](#_Toc523675102)

[3-5-1 درس­ها و سطرهای رنگی 24](#_Toc523675103)

[3-5-2 الگوریتم های مرحله2 از گام کاستن2 25](#_Toc523675104)

[3-6 الگوریتم پیشنهاد برنامه­های ترمی 27](#_Toc523675105)

[3-6-1 زیرساخت اول 27](#_Toc523675106)

[3-6-2 بخش اصلی الگوریتم 31](#_Toc523675107)

[3-7 ارائه ترتیبی قابل اخذ برای سطرهای یک برنامه ترمی 35](#_Toc523675108)

[3-8 خلاصه 35](#_Toc523675109)

**فصل چهارم:نسخه آزمایشی و نتایج آزمایشی**

[4-1 مقدمه 37](#_Toc523675110)

[4-2 آشنایی با نسخه آزمایشی 37](#_Toc523675111)

[4-3 نتایج حاصل از آزمایشات 44](#_Toc523675112)

[4-4 خلاصه 45](#_Toc523675113)

**فصل پنجم: نتیجه گیری و پیشنهادات**

[5-1 نتیجه گیری­ها 47](#_Toc523675114)

[5-2 پیشنهادها 47](#_Toc523675115)

فهرست تصویر

عنوان صفحه

[شکل 2-1 دید ما از یک الگوریتم در حالت کلی 7](#_Toc523675713)

[شکل 2-2. سطرهایی از جدول دروس ارائه شده در یکی از ترم­ها 8](#_Toc523675714)

[شکل 2-3 تشخیص نحوه برخورد الگوریتم با ورودی­ها از روی خروجی ها 8](#_Toc523675715)

[شکل 2-4 شمای کلی از مولفه­های اصلی پروژه 13](#_Toc523675716)

[نمودار 2-5. تصویر کلی از مولفه­های اصلی پروژه 14](#_Toc523675717)

[تصویر 3-1ساختار توصیف کننده برنامه درسی وزارت علوم مصوب سال 92 برای رشته مهندسی کامپیوتر 18](#_Toc523675718)

[شکل 2-3 مدل گرافیکی نمونه­ای از ارتباطات Course،OfferedCourse،OfferedCourseRow و Main.OfferedCourseRow 28](#_Toc523675719)

[شکل 3-3 شیوه محاسبه کردن تمام برنامه­های ممکن و مجاز برای یک مجموعه درس ورودی سه تایی 29](#_Toc523675720)

[شکل 4-1. تصویری از صفحه خانه نرم­افزار 37](#_Toc523675721)

[شکل 4-2 تصویری از قسمت­های تعین گرایش و تمرکز تخصصی 38](#_Toc523675722)

[شکل 4-3 تصویری از بخش جدول تطبیق دروس 38](#_Toc523675723)

[شکل 4-4 تصویری از بخش تاریخچه دانشجو 39](#_Toc523675724)

[شکل 4-5 تصویری از بخش بارگذاری دروس ارئه شده 39](#_Toc523675725)

[شکل 4-6 تصویری از نمونه فایل­های قابل دستیابی در سیستم گلستان 40](#_Toc523675726)

[شکل 4-7 تصویری از فایل­های ذخیره شده مربوط به دروس ارائه شده در ترم توسط دانشگاه 40](#_Toc523675727)

[شکل 4-8 تصویری از بخش بارگذاری دروس ارائه شده پس از تهیه موفقیت آمیز فایل هدف 41](#_Toc523675728)

[شکل 4-9تصویری از بخش تنظیمات پردازشی 41](#_Toc523675729)

[شکل 4-10 تصویری از بخش انتخاب واحد 42](#_Toc523675730)

[شکل 4-11تصویری از یک نمونه برنامه ترمی پیشنهاد شده توسط الگوریتم در نمایش لیست دروس 43](#_Toc523675731)

[شکل 4-12 تصویری از یک نمونه برنامه ترمی پیشنهاد شده توسط الگوریتم در نمایش جدولی 43](#_Toc523675732)

[شکل 4-13تصویری از بخش برنامه­های ذخیره شده 44](#_Toc523675733)

[شکل 4-14 تصویری از بخش تنطیمات 44](#_Toc523675734)

فصل اول:

مقدمه

1-1 موضوع پروژه

در این پروژه سعی شده است تا با تحلیل و بررسی و سپس توسعه برنامه ای کاربردی در قالب نمونه آزمایشی، راهکاری عملی برای اتوماسیون فرآیند انتخاب واحد که در ابتدای هر ترم توسط دانشجویان انجام می­شود ارائه گردد.

لازم به ذکر است که نسخه آزمایشی برای مهندسی کامپیوتر و مطابق برنامه درسی مصوب سال 92 وزارت علوم برای این رشته پیاده­سازی شده است. همچنین برای انجام این پروژه علاوه بر طراحی و پیاده­سازی الگوریتم­های اساسی مورد نیاز، لازم بود تا جزییات مهمی مانند محدودیت­ها و قوانین آموزشی، اولویت­های دانشجویان و ... در نظر گرفته شود تا پروژه نه فقط یک برنامه کاربردی بلکه یک برنامه کاربردی کاربر پسند باشد. غیر از اینها این دید را هم داشتم تا پروژه طوری پایه گذاری شود که برای رشته­های دیگر و سایر برنامه­های درسی مصوب وزارت علوم به راحتی قابل توسعه باشد.

1-2 اهمیت و کاربرد ها

موضوع این پروژه از آن جهت دارای اهمیت است که می­تواند فرآیند انتخاب واحد را برای دانشجویان تسریع و تسهیل نماید چرا که با پیشنهاد دادن تعدادی برنامه ترمی، مطابق با نیاز ها و اولویت های تعیین شونده توسط دانشجو، وی را قادر می­سازد تا با توجه به سلیقه خویش برنامه­­ای را جهت انتخاب واحد برگزیند و حال آنکه پیدا کردن چنین برنامه­ای بصورت دستی در عمل، معمولا کاری دشوارتر و اغلب زمانبرتر است.

علاوه بر این دانشگاه ها نیز می­توانند مشتریان این پروژه باشند و از آن برای اعمال تغییراتی در فرآیند انتخاب واحد ترمی دانشجویان استفاده کنند. برای مثال آنها می­توانند دانشجویان را قادر سازند تا پس از انتخاب برنامه، علاوه بر آنکه دانشجو می­تواند کد درس ها را بصورت دستی در سامانه ثبت کند، امکان ثبت اتوماتیک درس ها با توجه به برنامه انتخابی را نیز اضافه کنند.

1-3تعریف صورت مسئله بصورت دقیق تر

در ابتدای هر ترم، دانشگاه­ها برنامه­ای تحت عنوان **دروس ارائه شده در ترم** ارائه می­کنند که در واقع جدولی است که سطر های آن شامل مواردی از جمله درس ، دانشکده ارائه دهنده درس ، استاد ارائه دهنده درس ، زمان و مکان برگزاری کلاس ها و تاریخ امتحان ، ظرفیت و ... می­باشد و دانشجو در ابتدای هر ترم باید با توجه به این جدول برنامه ای را برای ترم خود برگزیند.

هر دانشجو علاوه بر اینکه خود را ملزم می­داند تا برنامه­ای که انتخاب می­کند مطابق با قوانین، از جمله قوانین مصّوب وزارت علوم و آئین نامه های داخلی دانشگاه باشد، به دنبال برنامه ای است که با اولویت ها و سلیقه وی مطابق باشد.

بنابراین توضیح صورت مسئله بصورت دقیق­تر این است که ما می­خواهیم دانشجو را در یافتن برنامه ترمی مورد نظرش یاری کنیم و بجای آنکه دانشجو بطور دستی بخواهد برنامه ای را برای خود برگزیند ما بر اساس اولویت های وی برخی از برنامه های ترمی مجازی که محدودیت های قانونی را ارضا می­کنند پیدا کنیم و تعدادی از بهترین برنامه هایی را که در بازه زمانی تعیین شده­ پیدا کرده ایم به او ارائه دهیم.

تا به اینجا ما **صورت مسئله** را بصورت دقیق­تر بیان کردیم اما این صورت مسئله همچنان ابهام دارد چرا که ما هنوز محدوده پروژه را مشخص نکرده­ایم و به همین دلیل از محدودیت هایی که با توجه به حدود پروژه مشخص می­شوند غافل مانده­ایم.

ما محدوده پروژه را بدین صورت تعریف می­کنیم که، پروژه ما با توجه به سیستم **گلستان** که در حال حاضر یکی از سامانه های مورد اقبال در زمینه سیستم های جامع آموزشی در میان دانشگاه های کشور محسوب می­شود و برای رشته های فنی و مهندسی و براساس قوانین دانشگاه صنعتی نوشیروانی بابل انجام شود.به علاوه مسئله فقط لازم است برای دانشجویانی که تنها در یک رشته تحصیلی مشخص تحصیل می­کنند و برای دوره کارشناسی حل شود.

با مشخص شدن حدود پروژه محدودیت های جدیدی به پروژه اعمال می­شوند برای مثال گفته شد پروژه قرار است برای سیستم گلستان نوشته شود و در حال حاضر در هنگام انتخاب واحد در این سیستم شما باید واحد ها را به ترتیبی ثبت کنید که همواره برنامه ترمی شما در وضعیت معتبری قرار داشته باشد یعنی برای مثال جهت اخذ درسی که همنیازی دارد ابتدا باید همنیاز آن درس را وارد کرده سپس خود درس را وارد کنید بنابراین به صورت مسئله این خواسته بصورت ضمنی اضافه می­شود که علاوه بر پیشنهاد برنامه ترمی، درس های آن برنامه پیشنهادی را با ترتیبی قابل اخذ برای سیستم گلستان به نمایش درآورد.

بنابراین آنچه پیش­­تر بیان شد به علاوه محدودیت هایی که حدود پروژه را تعیین می­کند صورت مسئله را بصورت دقیق تعریف می­کنند.

1-4 اهداف پروژه

ما می­خواهیم مسئله­ای که پیش­تر در آخر 3-1 بصورت دقیق تعریف کردیم را تحلیل کرده و یک برنامه کاربردی در قالب نسخه­ای آزمایشی که بتواند مسئله را بصورت کارا حل کند پیاده­سازی کنیم. همچنین ما می­خواهیم در تحلیل و حتی­الامکان در پیاده­سازی خود به گونه­ای عمل کنیم که این پروژه به جای آنکه به محدودیت های تعیین شده توسط حدود پروژه وابسته باشد، این محدودیت ها را ارضا کند اما تحلیل و پیاده­سازی به گونه­ای باشد که بتوان پروژه را برای محدوده های دیگر به راحتی و به سرعت انجام داد برای مثال با صرف زمانی اندک یا حتی بدون صرف هیچ زمانی بتوان آن را برای دانشگاه دیگری پیاده­سازی کرد یا بتوان آن را برای برنامه های درسی وزارت علوم که در آینده می­آیند، به سرعت پیاده­سازی کرد.

1-5 گام­های انجام پروژه

پروژه­ای که با آن سر و کار داریم یک پروژه نرم­افزاری است و برای انجام یک پروژه نرم­افزاری بصورت کلی می­توانیم گام های زیر را در نظر بگیریم:

* نیاز های پروژه را تشخیص داده و پروژه را تحلیل کنیم.
* به طراحی و پیاده سازی نرم­افزار بپردازیم.
* به بازبینی و تست نرم­افزار و ارزیابی تست بپردازیم.

مواردی که در بالا به آن اشاره کردیم بسیار کلی هستند و علیرغم آنکه تقریبا برای هر پروژه نرم­افزاری لازم و ضروری می­باشند اطلاعات ملموسی در اختیار ما نمی­گذارند همچنین از ترتیب انجام کار ها نیز اطلاعاتی در اختیار نمی­گذارند بنابراین فکر کردم بهتر است بصورت تیتر وار مراحلی که واقعا برای انجام این پروژه طی شده است را ذکر کنم. این مراحل به ترتیب شامل موارد زیر می­باشند:

* مطالعه و بررسی جدیدترین برنامه های درسی فنی مهندسی مصوب وزارت علوم و تحلیل کلی پروژه که طی آن داده های ورودی مسئله و همچنین مراحل لازم برای انجام مسئله از قبیل گام کاستن1، گام کاستن2، الگوریتم پیشنهاد برنامه ترمی و امکانات بیشتر بدست آمدند بعلاوه مشخص کردن تکنولوژی های مورد استفاده برای پیاده­سازی و بررسی معماری های مختلف نیز در این بخش انجام شد.
* تهیه داده های اولیه - پیاده­سازی بخش فراهم کننده خدمات فایل ها و مدلسازی دروس ارائه شده در ترم مبتنی بر سیستم گلستان - بعلاوه تست بخش پیاده­سازی شده.
* مطالعه مجدد برنامه های درسی وزارت علوم بعلاوه برنامه های تازه تصویت شده در فاصله این بخش و بخش دوره قبل با عمق بیشتر – تحلیل و بررسی این برنامه ها و تلاش برای ارائه ساختاری عمومی که بتواند برنامه های درسی وزارت علوم را مدل کند – پیاده­سازی ساختار ایجاد شده و تست آن.
* پیاده­سازی و تست بخش مقدماتی نرم­افزار – پیاده­سازی و تست گام های کاستن.
* تحلیل و پیاده­سازی و تست امکانات بیشتر شامل نمایش جدولی برنامه های پیشنهاد شده توسط نرم­افزار و نمایش آنها با ترتیبی قابل اخذ برای سیستم گلستان بعلاوه امکان افزودن اطلاعات جانبی برای هر برنامه ترمی پیشنهادی و ذخیره سازی آن.
* تهیه داده های جامع برای تست و ارزیابی نرم­افزار.
* طراحی الگوریتم برای جستجو و یافتن و سپس پیشنهاد دادن برنامه های ترمی مناسب – پیاده­سازی و تست الگوریتم.
* پیاده­سازی بخش های **خانه** و **تنظیمات** و تست آنها.
* ارزیابی نرم­افزار، تشخیص عیوب و سپس بازنگری و اصلاح کد بعلاوه تغییر بعضی از نامگذاری ها.

ارزیابی مجدد نرم­افزار.

ما فعلا از توضیحات فنی که دقیقا هر کدام از این موارد ذکر شده چه هستند صرف نظر می­کنیم و در بخش های بعدی با این که هر کدام از این موارد دقیقا چه هستند بیشتر آشنا می­شویم.

همانطور که در بالا ذکر شده است گام­های انجام پروژه مانند آنچه در ابتدا بصورت کلی و در سه بخش ارائه شده بود قابل تفکیک و جدا از هم نیست بلکه بنا به فراخور هر مرحله برای انجام آن مرحله اقدام شده­است با این وجود از آنجا که نوشتن گزارشی بر مبنای ده موردی که به آنها اشاره شده بود کاری دشوار است ما فصل­بندی ها را بگونه­ای قرار می­دهیم که فهم پروژه را تسهیل کند و توضیح و تشریح آن آسان تر گردد.

1-6 خلاصه

در این فصل ابتدا پروژه را بصورت کلی معرفی کردیم و بعد از آن گفتیم که چرا این پروژه می­تواند حائز اهمیت باشد سپس به تعریف دقیق مسئله­ای که این پروژه قصد حل کردن آن را دارد پرداختیم و پس از برشمردن اهداف پروژه، گام­های­ انجام آن را ابتدا در حالت کلی و سپس در حالت واقعی بیان کردیم و پس از مقایسه این دو حالت، دلایلی آوردیم که شیوه تنظیم فصل­های ما را روشن می­کند.

فصل دوم:

طراحی اولیه و آشنایی با مفاهیم پایه

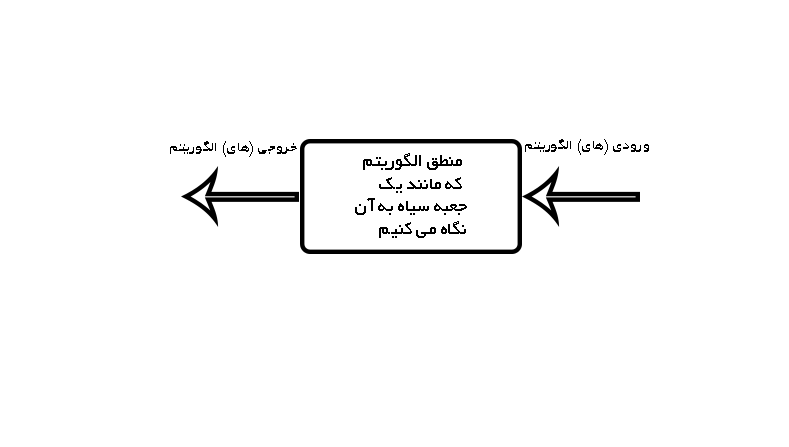
2-1 مقدمه

شرح کامل و جز به جز و طراحی پروژه می­تواند کاری خسته کننده باشد چرا که در فرآیند طراحی اغلب به جزئیاتی برمی­خوریم که برای دقیق­تر بودن پروژه نیاز است آنها را بررسی کنیم اما در عمل و پس از اتمام این فرآیند متوجه می­شویم به بسیاری از این جزئیات نیازی نداشته­ایم. از این جهت ما در اینجا به جای آنکه بخواهیم جز به جز آنچه بررسی کرده­ایم را بیان کنیم، به تشریح نتایج حاصل از طراحی می­پردازیم چرا که آنچه که عملا با آن سر و کار خواهیم داشت آنها هستند.

2-2 فشرده اول از طراحی

پس از تفکر پیرامون مسئله ای که در این پروژه قصد حل کردن آن را داشتیم متوجه شدیم که قلب این پروژه طراحی الگوریتمی جهت پیشنهاد برنامه ترمی می­باشد.

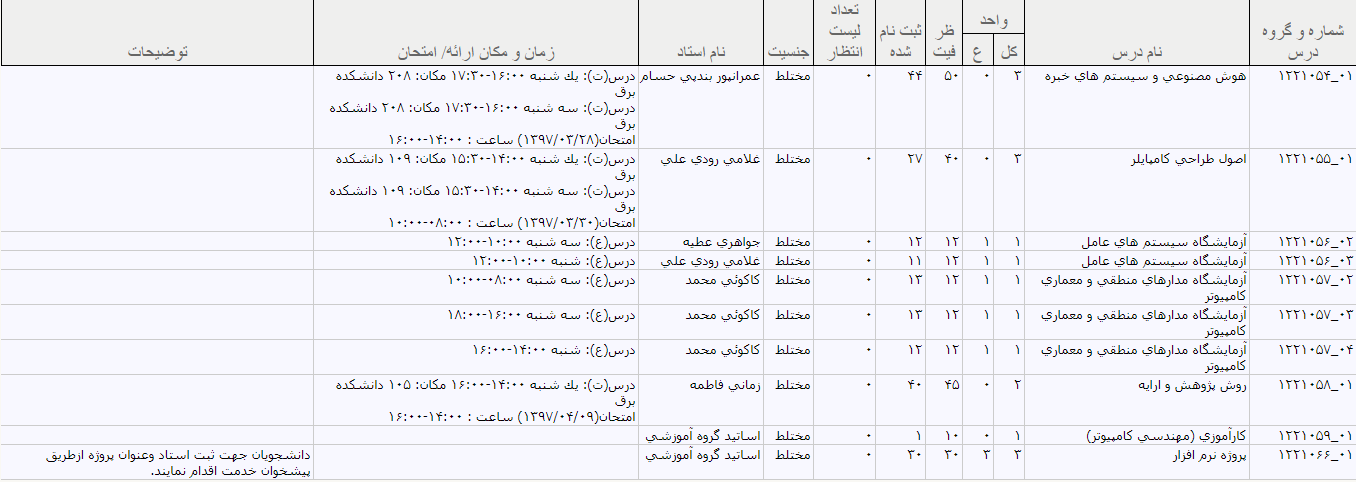
از آنجا که هر الگوریتم در حالت کلی به ترتیب شامل سه قسمت : ورودی ، منطق الگوریتم و خروجی آن می­باشد، سعی کردیم صرف نظر از جزیئات تصور کنیم که ورودی و خروجی الگوریتم پیشنهاد برنامه ترمی باید چگونه باشد.



شکل 2-1 دید ما از یک الگوریتم در حالت کلی

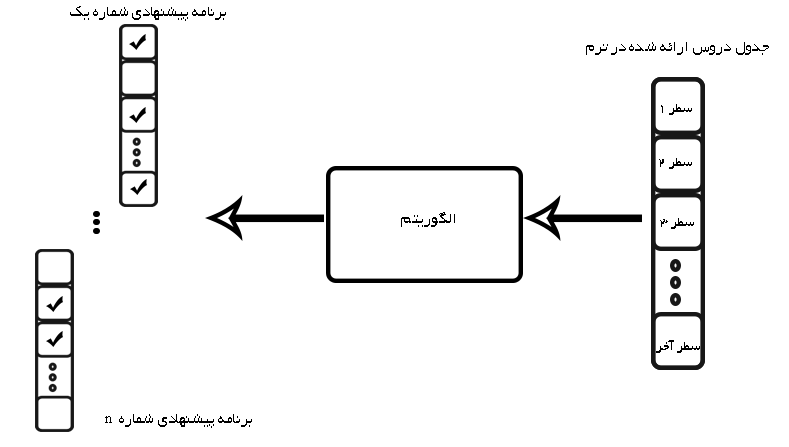
در واقع منطق الگوریتم را مانند جعبه سیاهی در نظر گرفتیم که فعلا نمی­دانیم که قرار است چگونه باشد اما با مشخص کردن ورودی ها و خروجی های حاصل از آن می­خواهیم عملکرد و تاثیری که این جعبه سیاه در خروجی هایش می­گذارد را شناسایی کنیم.

در فصل قبل گفتیم **دروس ارائه شده در ترم** به صورت یک جدول، هر ترم توسط دانشگاه ارائه می­گردد و این جدول شامل سطر هایی شامل مواردی از جمله درس ، دانشکده ارائه دهنده درس ، استاد ارائه دهنده درس ، زمان و مکان برگذاری کلاس ها و تاریخ امتحان ، ظرفیت و ... می­باشد. در شکل زیر می­توانید نمونه ای واقعی از سطر هایی از این جدول را که در سیستم گلستان ارائه شده است مشاهده کنید.



شکل 2-2. سطرهایی از جدول دروس ارائه شده در یکی از ترم ها

ما همه­ی ورودی ها و خروجی­ های الگوریتم را در این مرحله تعیین نمی­کنیم اما بدیهی است که یکی از ورودی های اصلی برای مسئله ما جدول **دروس ارائه شده در ترم** می­باشد و الگوریتم باید در خروجی تعدادی از سطر های این جدول را که می­خواهد به عنوان برنامه پیشنهاد ­دهد برگرداند. با این اوصاف می­توانیم هر سطر از این جدول را مانند جعبه سیاهی در نظر بگیریم که الگوریتم در مورد انتخاب شدن یا انتخاب نشدن آن سطر در هر برنامه پیشنهادی­ باید تصمیم گیری کند.



شکل 2-3 تشخیص نحوه برخورد الگوریتم با ورودی ها از روی خروجی ها

اما نکته مهم ­این است که تعداد سطر­هایی که در جدول دروس ارائه شده در ترم وجود دارند نسبت به آنچه دانشجو مجاز به اخذ آنها در آن ترم است بسیار بیشتر است. این نکته برای ما الهام بخش ایده­ای بود که آن را **کاستن** نام نهادیم.

ایده کاستن به این معنی است که ما سعی کنیم از فضای بزرگ مسئله بکاهیم و آنرا کوچک تر کنیم.در واقع این ایده سعی می­کند برخی از سطر­های جدول دروس ارائه شده در ترم را، که با توجه به شرایط دانشجو غیر قابل اخذ می­باشند قبل از ارسال برای الگوریتم از ورودی حذف کند.

2-3 فشرده دوم از طراحی

مشاهده کردیم که ایده کاستن چگونه می­تواند با تلاش برای کوچک کردن فضای مسئله، فضای جستجو برای الگوریتم را کوچک تر کند. اما تا به اینجا ایده کاستن را توصیف کردیم و هنوز نگفتیم که این ایده قرار است چگونه در عمل بکار گرفته شود.در ادامه به تشریح این مقوله می­پردازیم.

2-3-1 گام کاستن 1

هر دانشجو شرایطی دارد که با توجه به آن ممکن است دامنه سطر­های قابل اخذ جدول برنامه دروس ارائه شده در ترم برایش محدود تر شود.برای مثال باید سطر­هایی که برای جنسیت زن بکار برده شده است را از دامنه سطر­های قابل اخذ برای دانشجو مرد حذف کرد.

دیدیم که یکی از راه های کاستن استفاده از شرایط دانشجو است. ما پس از تحلیل و بررسی بیشتر، شرایطی از دانشجو را که می­توان از آن برای کاستن استفاده کرد مشخص کردیم و از میان آنها آن شرایطی را که می­خواستیم در نرم­افزار پیاده­سازی کنیم مشخص کردیم.

در نهایت شرایطی را که برای پیاده­سازی انتخاب کردیم شامل جنسیت، شماره ترمی که دانشجو در آن می­خواهد انتخاب واحد کند(یعنی دانشجو ترم چند است)،تاریخچه دانشجو و تعداد واحد گذرانده شده توسط دانشجو (که این یک خصوصیت محاسباتی است و می­توان آن را با توجه به تاریخچه دانشجو محاسبه کرد) می­باشد.

ما تمام شرایط موجود دانشجو را انتخاب نکردیم برای مثال در سیستم گلستان سطر­هایی وجود دارند که مشخص می­کنند آن سطر خاص فقط برای ورودی های خاصی مثلا ورودی های 91 و قبل از آن قابل اخذ می­باشد و ما چنین شرطی را در لیستی که پیشتر گفتیم انتخاب نکردیم در این رابطه باید به چند نکته مهم توجه کرد. نکته اول آنکه برای این مثال خاص باید بدانید که سیستم گلستان برای هر سطر یک قسمت مربوط به توضیحات دارد و از آنجا که توضیحات احتمالا می­توانند هر چیزی باشند ما در حالت کلی نمی­توانیم توضیحات را توسط نرم­افزار پردازش کنیم مگر آنکه به برنامه قابلیت پردازش زبان طبیعی بدهیم و از آنجا که چنین موردی در دامنه فعلی پروژه ما نمی­گنجید آن را انتخاب نکردیم. نکته دوم اینکه با این حال موردی که ذکر شد جز مواردی است که باید حتمی حذف سطر و کاستن برایش اتفاق بیفتد چرا که اگر در برنامه پیشنهادی الگوریتم چنین سطری موجود باشد، آن برنامه پیشنهادی نامعتبر خواهد بود ما برای مواجه با این مسئله دو راه حل داشتیم اولین راه­حل آنکه هر چند قصد نداریم زبان طبیعی را پردازش کنیم اما این نوع توضیحات در سیستم گلستان و در دانشگاه صنعتی نوشیروانی بابل تا به اینجا اغلب و یا شاید همیشه با الگو­های خاص قابل تشخیصی نوشته شده­اند بنابراین راه­ اول ما می­تواند این باشد که این الگو ها را شناسایی کنیم و پردازش را به همین الگو های شناسایی شده محدود کنیم اما همانطور که در فصل اول قسمت اهداف پروژه ذکر شده بود ما نمی­خواهیم پروژه خود را وابسته به محدوده مسئله کنیم و حال آنکه چنین راه­کاری دقیقا یکی از مواردی است که می­تواند قسمتی از پروژه را وابسته به محدوده مسئله کند. در عوض ما راه­کار دیگری را برگزیدیم که در آن این نوع تشخیص بر عهده کاربر قرار می­گیرد که در ادامه با آن آشنا خواهید شد.اما نکته سوم یک تاکید بیشتر است بر این موضوع که توجه داشته باشید که ما گفتیم می­خواهیم **سعی**(سعی با انجام دادن حتمی کار بصورت کاملا موفق متفاوت است)کنیم از فضای حالات کلی بکاهیم و کاستن در اینجا به معنی کوچک تر کردن فضای مسئله با حذف **برخی** از سطر­های غیر مجاز است و نباید به اشتباه تصور کرد که الزاما آنچه بعنوان ورودی به الگوریتم داده می­شود صرفا حالات مجاز است.

ما پیشتر از لفظ **تاریخچه دانشجو** استفاده کردیم مقصود ما از تاریخچه دانشجو اطلاعات مربوط به دروسی است که دانشجو آنها را با موفقیت گذرانده یا آنها را اخذ کرده اما نتوانسته با موفقیت بگذراند. ما با استفاده از اطلاعات حاصل از لیست انتخاب شده برای شرایط دانشجو به خصوص از تاریخچه دانشجو و با توجه به برنامه درسی وزارت علوم که دانشجو با آن در ارتباط است می­توانیم متوجه شویم دانشجو در یک ترم خاص مجاز است چه درس هایی را در صورت ارائه شدن اخذ نماید.

از این پس ما به فرآیند حذف کردن سطر­هایی از جدول برنامه دروس ارائه شده در ترم که با استفاده از شرایط انتخاب شده دانشجو شامل جنسیت، شماره ترمی که دانشجو در آن می­خواهد انتخاب واحد کند،تاریخچه دانشجو و تعداد واحد گذرانده شده توسط دانشجو بعلاوه برنامه و اطلاعات مخصوص مربوط به برنامه درسی مصوب وزارت علوم که مرتبط با دانشجو می­باشد، **گام کاستن 1** می­گوئیم.

2-3-2 گام کاستن 2

گام کاستن 1 تلاش لازمی بود که باید نهایتاً در قسمتی از پروژه اگر نه خود آن ولی حداقل عملکرد آن اعمال می­شد با این حال این گام تنها گامی نیست که می­توان در جهت کاهش فضای حالات برداشت. در این قسمت با گام کاستن 2 آشنا می­شوید که گامی است در حالت کلی، اختیاری ولیکن ممکن است در شرایطی به حالت اجبار نیز در ­آید.

شاید پیدا کردن یک برنامه ترمی مجاز با توجه به جدول برنامه دروس ارائه شده در ترم اغلب کار آسانی باشد اما کار زمانی سخت می­شود که پای الویت های محدود کننده دانشجو به میان می­آید. بنابراین پیش از آنکه بگوئیم گام کاستن 2 چیست می­خواهیم بدانیم الویت های دانشجویان برای انتخاب برنامه ترمی مورد نظرشان چگونه است.

دانشجویان برنامه ترمی خود را بر اساس الویت هایی انتخاب می­کنند که ما این اولیت ها را به دو دسته کلی **اولویت‌های باید** و **اولویت های شاید** دسته­بندی کردیم. الویت های باید الویت هایی هستند که باید در برنامه ترمی دانشجو **الزاما** ارضا شوند درحالیکه الویت های شاید الزام آور نیستند و می­توانند در حالت کلی با سه جمله **بهتر است این گونه باشد** ، **بهتر است اینگونه نباشد** و **فرقی ندارد اینگونه باشد یا نباشد** توصیف شوند در کنار دو نوع الویت باید و شاید، الویت دیگری بنام **نباید** وجود دارد اما از آنجا که می­توان نباید را با نقیض باید توصیف کرد از قرار دادن آن به عنوان یک دسته جداگانه از الویت ها خودداری کردیم.

رعایت قوانین از جمله قوانین وزارت علوم و آئین نامه های داخلی دانشگاه از اولویت های **باید** به حساب می­آیند اما اولویت های باید الزاما محدود به رعایت قوانین نیستند بلکه اولویت های **باید** می­توانند بر اساس آگاهی دانشجو از اموری **حتمی** ولی **غیر قابل مشاهده** برای ما(نرم­افزار) الزام آور شوند.

می­توان منشا اولویت های دانشجو را از **اهداف استراتژیک** وی دانست. اهداف استراتژیک اهدافی هستند که دانشجو با توجه به آنچه خود برنامه ریزی و هدف گذاری می­کند مشخص می­شوند برای مثال ممکن است دانشجویی بخواهد همزمان با تحصیل کار نیمه وقتی داشته باشد و بنابراین زمان های خاصی را الزاما در برنامه ترمی خود بخواهد خالی نگه دارد یا مثلا دانشجویی با این هدف که هر چه زود تر فارغ­التحصیل شود بخواهد هر ترم حداکثر تعداد مجاز واحد را اخذ کند یا بعنوان مثالی دیگر دانشجویی درس خاصی را بخواهد حتما با استاد خاصی بگیرد یا نگیرد و مثال های این چنینی دیگر که بسیارند.بنابراین برنامه ما باید این قابلیت را داشته باشد که دانشجو بتواند با استفاده از آن اولویت های مد نظرش را برای نرم­افزار مشخص کند.

پس از بحث بالا در این نوبت باید مشخص کنیم که نرم­افزار ما چه اولویت هایی را می­خواهد پوشش دهد. برای رسیدن به این هدف ابتدا فهرستی از برخی نتایج معمول حاصل از اهداف استراتژیک دانشجویان به شرح زیر تهیه کردیم:

* مشخص کردن محدوده واحد (مثلا این که دانشجو بخواهد برنامه ترمی وی بین 16 تا 19 واحد باشد)
* اخذ کردن یک یا چند درس بصورت باید(دانشجو بخواهد درس یا درس هایی را در ترم جاری حتما اخذ کند)
* اخذ نکردن یک یا چند درس بصورت نباید(دانشجو نخواهد در هر حال یک یا چند درس خاص را در ترم جاری اخذ کند.)
* اخذ کردن یک گروه درسی خاص(معادل یک سطر از جدول برنامه دروس ارائه شده در ترم) از یک درس بصورت باید(مثلا از بین چهار گروه ارائه شده برای درس سیگنال و سیستم دانشجو بخواهد حتما یک سطر خاص از این درس اخذ شود)
* اخذ نکردن یک یا چند گروه خاص از یک درس به هیچ وجه.
* اخذ کردن یکی از چند درس مشخص بصورت باید(مثلا دانشجو بخواهد بین دو درس از گروه معارف حتمی یک درس را از بین آنها در ترم جاری اخذ کند)

علاوه بر موارد بالا می­توان برای بعضی از آنها حالت **شاید** نیز اضافه کرد (برای مثال اینکه بگوییم این درس بهتر است در این ترم اخذ شود یا نشود) همچنین احتمالا علاوه بر همه این ها بتوان موارد دیگری نیز به این فهرست افزود.

اگر بخواهیم از نگاه توانمندی کاربر به این فهرست نگاه کنیم هر چه این فهرست شامل موارد جزئی­تر شود احتمالا کاربر در توصیف اولویت هایش توانا تر خواهد بود اما دو نکته از نظر تحلیل و طراحی وجود دارد اول آنکه زیاد شدن بیش از اندازه موارد می­تواند بجای آنکه کار کاربر را آسان کند عملا باعث سختی بیشتر کار او شود چرا که قبل از آنکه کاربر بتواند از امکانات نرم­افزار استفاده کند باید بداند که آنها چیستند و چگونه کار ­می­کنند. نکته دوم اینکه از دید تحلیلی فقط توصیف اولویت های باید می­توانند به کاستن فضای حالات مسئله بصورت مستقیم کمک کنند چرا که الویت های **شاید** تنها درصدی از اقبال و علاقه کاربر نسبت به سطرهای جدول برنامه دروس ارائه شده در ترم را بیان می­کنند و در نتیجه نمی­توان آنها را از جدول حذف کرد.با این اوصاف ما باید امکاناتی را برای پیاده سازی انتخاب کنیم که هم بتواند کاربر را در توصیف اولویت هایش توانا کند و هم باید از زیاد شدن بیش از اندازه امکانات بگونه ای که عملا کار را برای کاربر دشوار کند پرهیز کنیم، غیر از آن نباید این امکانات بگونه ای کم یا غیر کاربردی باشند که کاربر تمایلی به استفاده از آنها نداشته باشد(توجه داریم که گام کاستن 2 در حالت کلی اختیاری است) علاوه بر همه این ها تمایل داریم امکاناتی را انتخاب کنیم که از نظر پیاده­سازی نیز ساده­تر باشند. با بررسی کردن همه این موارد به این نتیجه رسیدیم که پنج مورد اول از فهرست پیش­تر بیان شده را پیاده­سازی کنیم. در واقع برای گزینه های شاید فقط حالتی که با **فرقی ندارد اینگونه باشد یا نباشد** بیان می­شود را باقی گذاشتیم و مورد ششم از لیست ذکر شده را صرفا برای راحت تر شدن پیاده­سازی و همچنین بخاطر ترس از پیچیده شدن بیش از اندازه الگوریتم از لیست انتخابی حذف کردیم.

اما برای اینکه بتوانیم با توجه به اولویت های کاربر در مورد مواردی از قبیل **ممکن بودن** اولویت کاربر ، متناقض نبودن آن با سایر الویت هایش آگاهی یابیم و همچنین برای تصمیم گیری در مورد حذف کردن برخی از سطر ها جهت کاستن از فضای حالات، باید بدانیم که در بررسی هایمان کلاس هایی که ظرفیت آنها پُر شده است را فیلتر کنیم یا فیلتر نکنیم بعلاوه باید بدانیم تداخل امتحانی نیز چک شود یا نشود. پس باید این دو مورد نیز از کاربر پرسیده شود.

از این پس به فرآیند دریافت اطلاعات و اولویت های کاربر و تلاش برای حذف سطر­هایی از جدول برنامه دروس ارائه شده در ترم با توجه به اولویت های مشخص شده، **گام کاستن2** می­گوئیم.

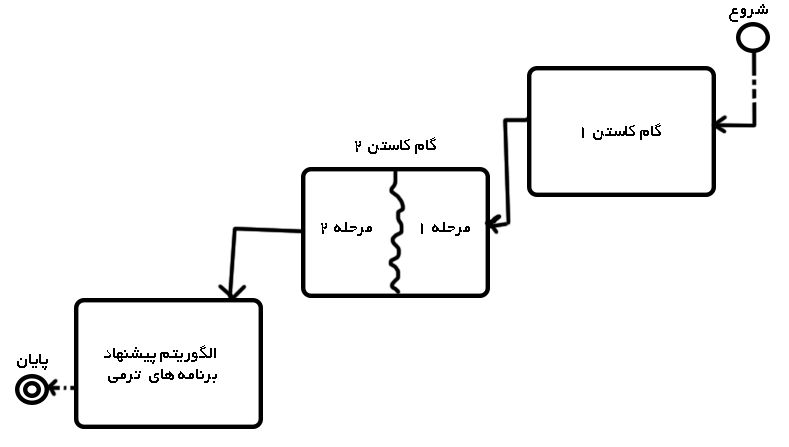
همچنین به فرآیند دریافت الویت ها و اطلاعات کاربر شامل حداقل و حداکثر واحد برای اخذ در ترم جاری،بررسی یا عدم بررسی تداخل امتحانات و فیلتر کردن یا فیلتر نکردن سطر­های با ظرفیت پُر و تلاش برای حذف سطر­هایی از جدول برنامه دروس ارائه شده در ترم با توجه به اولویت های مشخص شده، **مرحله1 از گام کاستن2** می­گوئیم.

همینطور به فرآیند دریافت الویت های کاربر بجز آنچه در مرحله1 از گام کاستن2 مشخص شد و تلاش برای حذف سطر‌هایی از جدول برنامه دروس ارائه شده در ترم با توجه به اولویت های مشخص شده، **مرحله2 از گام کاستن2** می‌گوئیم.

قبل از به پایان بردن این بخش می­خواهیم به قسمتی که در آن این سوال مطرح شده بود که در قبال ستون توضیحات سیستم گلستان چگونه عمل کنیم بازگشته، پاسخ دهم. اکنون اگر به لیست مواردی که برای گام کاستن2 مشخص کرده­ایم توجه کنید می­بینید می­توان براحتی با استفاده از این گام مشکل مربوط به ستون توضیحات را به کمک کاربر حل کرد و این یکی از همان شرایطی است که طی آن گام کاستن2 از حالت اختیاری خارج می­شود.

2-4 سایر مولفه ها و سازماندهی بخش های مربوط

با توجه به آنچه تا به اینجا گفته شد می­توان پروژه را پس از دریافت ورودی اولیه (یعنی ساختار توصیف کننده برنامه درسی مصوب از وزارت علوم که با دانشجو در ارتباط است) به ترتیب به صورت بخش­های گام کاستن1، مرحله1 از گام کاستن2 ،مرحله2 از گام کاستن2 و سپس الگوریتم پیشنهاد برنامه ترمی در نظر گرفت.



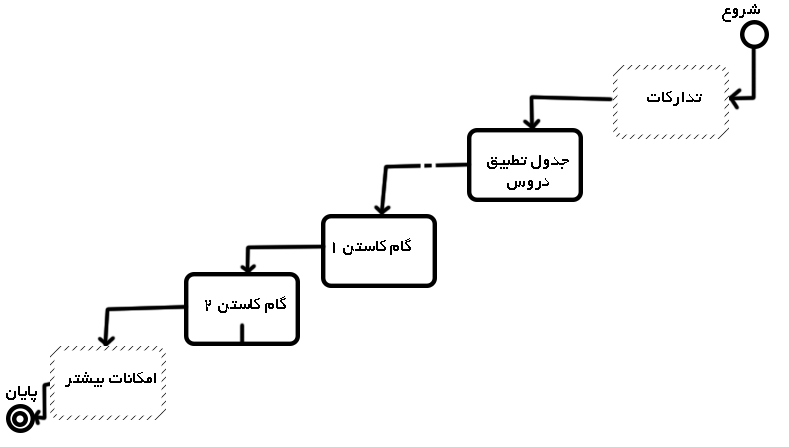
شکل 2-4 شمای کلی از مولفه های اصلی پروژه

حالا به طور فشرده به سایر مولفه ها می­پردازیم.

دانشگاه های مختلف الزاما تمام دروس موجود در برنامه درسی مصوب وزارت علوم را ارائه نمی­دهند به علاوه گاهی بجای یک درس موجود در برنامه مصوب وزارت علوم، درس معادلی را بجای آن ارائه می­دهند ما در مواجه با این موضوع مولفه­ای بنام **جدول تطبیق دروس** را به مولفه های نرم­افزار اضافه کردیم. این جدول شامل سطر­هایی با چهار ستون می­باشد که این ستون ها عبارتند از شناسه درس، عنوان درس در برنامه درسی مصوب وزارت علوم ، نام درس در دانشگاه مقصد و کُد درس در دانشگاه مقصد.

از این پس به درس­هایی از برنامه درسی مصوب وزارت علوم مرتبط با دانشجو که در دانشگاه مقصد(دانشگاهی که دانشجو با توجه به برنامه ارائه شده توسط آن دانشگاه انتخاب واحد می­کند)ارائه می­شوند **درس­های موجود** و به آنهایی که ارائه نمی­شوند **درس­های ناموجود** می­گوییم.

علاوه بر مولفه جدول تطبیق دروس، دو مولفه بنام های مولفه **تدارکات** و مولفه **امکانات بیشتر** را نیز اضافه می‌کنیم. در واقع این دو مولفه را به عنوان **جانگهدار** برای قسمت هایی که ممکن است در تحلیل دیده نشده باشند به مولفه‌های پروژه افزودیم.



نمودار 2-5. تصویر کلی از مولفه های اصلی پروژه

پس از پیاده­سازی، در نمونه آزمایشی پروژه، مولفه تدارکات شامل بخش­های مربوط به **خانه** و **تعیین گرایش و تمرکز** شد و مولفه امکانات بیشتر نیز شامل بخش­های **نمایش و** **افزودن اطلاعات بیشتر به برنامه های پیشنهادی و ذخیره سازی آنها** و **تنظیمات** شد.

برای قسمت هایی از پروژه مثلا برای تبدیل فایل های برنامه دروس ارائه شده در ترم به فایل قابل پردازش برای نرم­افزار به مولفه­ای مستقل از سایر بخش ها نیاز داریم که خدمات مربوط به فایل­های مورد نیاز نرم­افزار را فراهم کند.این مولفه را در عمل پیاده­سازی می­کنیم اما آنرا در شکل ها نشان نمی­دهیم چرا که همانطور که گفتیم این بخش از سایر بخش ها مستقل پیاده­سازی می­شود.این مولفه را **فراهم کننده خدمات فایل ها** می­نامیم.

2-5 خلاصه

در این فصل نتایج بدست آمده از طراحی پروژه را تشریح کردیم. در ابتدا سعی کردیم تاثیر و عملکرد الگوریتم پیشنهاد برنامه ترمی را شناسایی کنیم. در همین میان با ایده­ای به نام کاستن آشنا شدیم و بعد از آن بخش قابل توجهی از این فصل را به تشریح این ایده پرداختیم و در پایان این فصل نیز مولفه­های دیگری از پروژه را شناسایی کرده­ایم و در نهایت مولفه­های پروژه را سازماندهی کردیم.

فصل سوم:

ساختارها، الگوریتم­ها و روش­های اصلی حل مسئله

3-1 مقدمه

در فصل قبل نتایج حاصل از تحلیل و طراحی پروژه را تشریح کردیم. در این فصل به مدلسازی داده­ها و ارائه الگوریتم ها و روش­های بکار گرفته شده برای حل مسائل اصلی­ پیش رو، می­پردازیم.

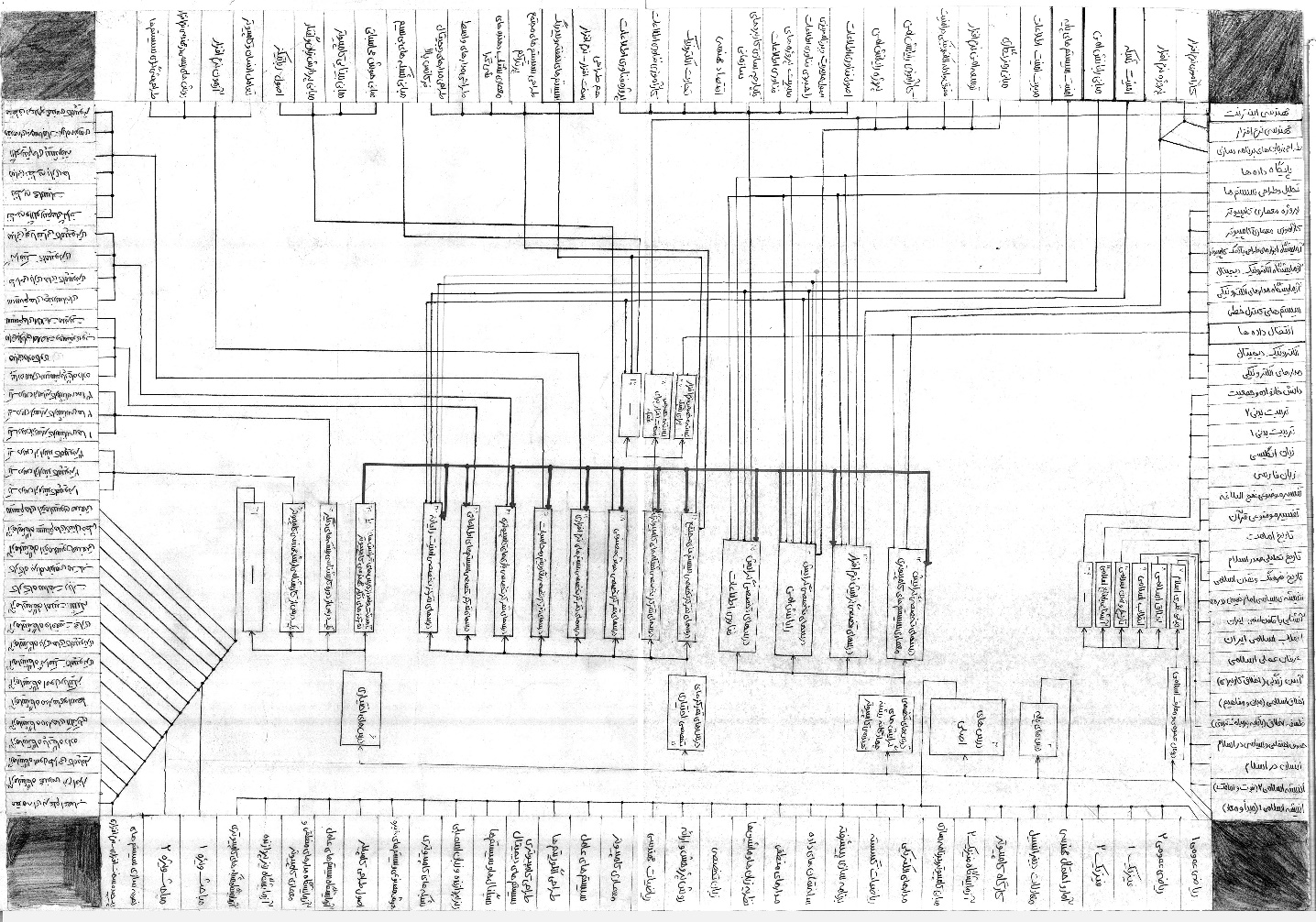
3-2 ساختار توصیف کننده برنامه درسی مصوب وزارت علوم

یکی از مهمترین ورودی­های مسئله ما برنامه درسی مصوب وزارت علوم است که با دانشجو در ارتباط است. این ارتباط ممکن است براساس مواردی از قبیل رشته تحصیلی، گرایش، سال ورود به دانشگاه و ... تعیین شود برای مثال دانشجوی مهندسی کامپیوتر ورودی 93 باید از برنامه درسی مصوب سال 92 وزارت علوم پیروی کند درحالیکه دانشجوی همین رشته اما ورودی 97، با برنامه درسی مصوب سال 96 آن، ارتباط دارد. تصمیم مهمی که باید می­گرفتیم درباره چگونه مدلسازی کردن برنامه درسی مصوب وزارت علوم بود. در ادامه به تشریح مدلسازی خود می‌پردازیم.

برنامه­های درسی مصوب وزارت علوم شامل دو مولفه **درس و محدودیت** می­باشند. درواقع هر برنامه درسی مصوب وزارت علوم شامل تعدادی درس است که وزارت علوم با محدودیت­هایی که در برنامه درسی مصوب خود اعمال می‌کند شرایط اخذ هر درس را مشخص می­کند. نکته دیگر اینکه این محدودیت­ها انواع شناخته شده و مشخصی دارند که عبازتند از: پیشنیازی و هم­نیازی دروس، حداقل ترم برای اخذ درس، حداقل تعداد واحد برای اخذ درس، حداقل و حداکثر واحد مجاز برای اخذ درس­های موجود در یک گروه از درس­ها یا انواعی از درس­های یک گروه(مثلا از درس­های پایه در برنامه مصوب 92 کامپیوتر باید 20 واحد گذرانده شود یا بعبارت دیگر حداقل و حداکثر 20 واحد)، به علاوه نوع دیگری از محدودیت­ها که روی ارتباط دانشجو با برنامه درسی مصوب وزارت علوم اعمال می­شود (مثلا برای دانشجو کامپیوتر ورودی 93 این نوع از محدودیت را می­توان، آن محدودیت هایی برای اخذ درس­ها درنظر گرفت که به واسطه گرایش و تمرکز تخصصی دانشجو مشخص می­شوند). می­توان سه محدودیت اول نامبرده شده را بعنوان خصیصه های یک درس در نظر گرفت. به این ترتیب مدل ما باید توانایی این را داشته باشد که سایر محدودیت­های تعریف نشده را پشتیبانی کند و همچنین با محدودیت­هایی که بعنوان خصیصه به درس­ها انتساب داده شده­اند سازگار باشد. با این اوصاف برای ساختار خود عناصری تعریف کردیم که با استفاده از آنها بتوان برنامه مصوب وزارت علوم را به شکل یا شکل­هایی توصیف کرد.این عناصر عبارتند از:

* درس -> یک درس منبعی است که می­تواند به یک یا چند گروه وصل باشد(ارتباط داشته باشد).
* گروه یا دسته درسی -> یک گروه یا دسته درسی، عنصری است که می­تواند با دسته های درسی دیگر از طریق گیت­های ورودی یا خروجی ارتباط داشته باشد یا با درس های دیگر نیز ارتباط داشته باشد.
* کارت اعتباری -> ساختاری است که از آن برای تعیین اعتبار و تشخیص سطح دسترسی استفاده می­شود.
* محدودیت -> ساختاری است که با آن می­توان محدودیت­های حداقل و حداکثر واحد مجاز برای اخذ درس­های موجود در یک گروه از درس­ها یا انواعی از درس­های یک گروه را توصیف کرد.
* گواهینامه -> ساختاری است که می­تواند شامل تعدادی محدودیت بعلاوه سطوح دسترسی معین که به کمک کارت­های اعتباری قابل بیان است، باشد.
* گیت -> ساختاری است که می­تواند تعدادی گواهینامه را بر روی دسته درسی مبدا­ای به دسته درسی مقصدی مشخص کند.
* گیت ورودی و گیت خروجی -> گیت ورودی و گیت خروجی، مفاهیمی هستند که با توجه به جهت قراردادی، در حرکت از یک دسته درسی به دسته درسی دیگر شکل می­گیرند. برای مثال فرض کنید بین دسته درسی الف و ب از طریق یک گیت، ارتباطی برقرار باشد و جهت قراردادی نیز، از الف به ب باشد در این صورت مرجع گیت در الف گیت خروجی و مرجع گیت در ب گیت ورودی است.
* گروه یا دسته درسی فقط خروجی -> دسته درسی­ای که همه گیت­های آن با توجه به جهت قراردادی، خروجی باشند را گروه یا دسته درسی فقط خروجی می­نامیم.
* گروه یا دسته درسی فقط ورودی -> دسته درسی­ای که همه گیت­های آن با توجه به جهت قراردادی، ورودی باشند را گروه یا دسته درسی فقط ورودی می­نامیم.
* گروه یا دسته درسی هم ورودی هم خروجی -> دسته درسی­ای که با توجه به جهت قراردادی هم گیت­های ورودی و هم گیت­های خروجی داشته باشد را گروه یا دسته درسی هم ورودی هم خروجی می­نامیم.(در ساختار ما این نوعِ گروه نباید با هیچ درسی ارتباط باشد)
* ریشه -> گروه یا دسته درسی فقط خروجی را که ارجاع به هیچ درسی نداشته باشد ریشه می­نامیم.(هر ساختار توصیفی ما باید دقیقا یک ریشه داشته باشد.)
* گروه یا دسته درسی سطح اول -> گروه­ها یا دسته­های درسی­ای که با ریشه ارتباط داشته باشند را گروه­ها یا دسته­های درسی سطح اول می­نامیم.
* نود پایانی یا لبه -> گروه­ها یا دسته­های درسی فقط ورودی را که با حداقل یک درس در ارتباط باشند، نود پایانی یا لبه می­نامیم.

شاید درک این عناصر از روی توصیحات بالا کار دشواری باشد. در ادامه، کار خود را با یک مثال عملی ادامه می­دهیم.



تصویر 3-1ساختار توصیف کننده برنامه درسی وزارت علوم مصوب سال 92 برای رشته مهندسی کامپیوتر

تصویر پیشین، تصویری است از ساختار توصیف کننده برنامه درسی مصوب سال 92 وزارت علوم برای رشته مهندسی کامپیوتر. البته در این شکل همه عناصر ساختاری که پیش­تر معرفی کرده بودیم نمایش داده نشده اند برای مثال در این تصویر عناصر مربوط به گواهینامه نمایش داده نشده است.

در چهارگوشه تصویر عناصر **درس** قرار گرفته اند که هر کدام از روی عنوان نوشته شده روی آنها قابل تشخیص می­باشند.

جهت قراردادی ما از سمت **راست به چپ** می­باشد و بنابراین جعبه سیاه رنگی که بالا و سمت راست تصویر قرار دارد و گروهی است فقط خروجی، نمایانگر **ریشه** می­باشد.(سایر جعبه­های سیاه صرفا جنبه نمایشی دارند)

جعبه­هایی که در وسط صفحه­ قرار گرفته­اند(یعنی همه جعبه ها بجز درس­ها و سه جعبه تیره رنگ صرفا نمایشی)، **گروه**­ها هستند.

گروه­های دروس عمومی و معارف، درس­های تخصصی گرایش­های چهارگانه رشته مهندسی کامپیوتر، درس­های تمرکز تخصصی اختیاری و دروس اختیاری، درس­های تمرکز تخصصی شبکه­های کامپیوتری و تا هشت واحد از درس­های گرایش­ها یا تمرکز­های دیگر مهندسی کامپیوتر **گروه­های هم ورودی هم خروجی** هستند.

گروه­های دروس عمومی و معارف، درس­های پایه، درس­های اصلی، درس­های تخصصی گرایش­های چهارگانه رشته مهندسی کامپیوتر، درس­های تمرکز تخصصی اختیاری و دروس اختیاری گروه­های **سطح اول** هستند.

گروه­های درس­های پایه، درس­های اصلی، مبانی نظری اسلام، اخلاق اسلامی، انقلاب اسلامی، تاریخ و تمدن اسلامی، آشنایی با منابع اسلامی، خط تیره دروس عمومی و معارف اسلامی، درس­های تخصصی گرایش معماری سیستم­های کامپیوتری، درس­های تخصصی گرایش نرم­افزار، درس­های تخصصی گرایش رایانش امن، درس­های تخصصی گرایش فناوری اطلاعات، درس­های تمرکز تخصصی سیستم­های مجتمع، درس­های تمرکز تخصصی هوش مصنوعی، درس­های تمرکز تخصصی سیستم­های نرم­افزاری، درس­های تمرکز تخصصی الگوریتم و محاسبات، درس‌های تمرکز تخصصی بازی­های کامپیوتری، درس­های تمرکز تخصصی سیستم­های اطلاعاتی، درس­های تمرکز تخصصی امنیت رایانه، یک درس از دوره کارشناسی رشته­های دیگر، یک درس از کارشناسی ارشد مهندسی کامپیوتر، خط تیره دروس اختیاری، بسته مخصوص نرم­افزار برای شبکه، بسته مخصوص سخت­افزار برای شبکه و خط تیره درس‌های تمرکز تخصصی شبکه­های کامپیوتری گروه­های فقط ورودی هستند و **لبه**­های این ساختار را تشکیل می­دهند.

همانطور که قبل­تر نیز اشاره کردیم در این تصویر همه­عناصر نمایش داده­نشده­اند اما برای آنکه یک دید کلی از اینکه این عناصر چگونه و کجا هستند پیدا کنیم در این رابطه چند مثال ارائه می­دهیم- یالی که با گیت ورودی ریشه به گیت خروجی دروس اصلی مشخص شده را در نظر بگیرید، بر روی این یال گواهینامه­ای قرار دارد که طبق آن حداقل و حداکثر 59 واحد درسی ­می­تواند از آن عبور کند(محدودیت) و اخذ درس­های آن برای همه دانشجویان مجاز می­باشد(کارت اعتباری) در واقع فرض کنید یک دانشجو کامپیوتر که با این برنامه درسی ارتباط دارد با گرایش و تمرکز مخصوص به خود از ریشه شروع به حرکت به سمت لبه ها که منابع(درس های مختلف) آنجا قرار دارند می­کند، در این مسیر دانشجو باید بتواند از یال­ها بدون نقض کردن شرایط آنها عبور کند، مثلا اگر دانشجو تا به حال 3واحد از درس­های اصلی را گذرانده باشد می­تواند از ریشه به لبه­ی درس­های اصلی برود و از آنجا برای خود یکی از دروسی که مجاز به اخذ آنها است را انتخاب کند- بعنوان مثالی دانشجویی دیگر با گرایش مهندسی نرم­افزار که شش واحد از درس­های تخصصی و دو واحد آزمایشگاه از دروس اختیاری خود را گذرانده است در نظر بگیرید. در این صورت این دانشجو اگر از مسیر ریشه به درس­های تخصصی گرایش­های چهارگانه مهندسی کامپیوتر حرکت کند، فقط مجاز به اخذ درس از درس­های تخصصی گرایش نرم­افزار می­شود چرا که مجوز دسترسی به سه گروه تمرکز تخصصی دیگر را نداشته، تنها مجوز در این مسیر برای او مربوط به درس­های تمرکز تخصصی گرایش نرم­افزار می­باشد و از آنجا که تا به حال شش واحد از دروس این گروه را گذرانده بدون نقض کردن محدودیت­ها می­تواند به این گروه رسیده از آن درس یا درس­هایی مجاز برای خود اخذ کند. این در حالیست که همین دانشجو اگر بخواهد از ریشه به دروس اختیاری و سپس به دروس تخصصی گرایش نرم­افزار برسد مجوز لازم برای اینکار را نداشته ولی مجوز دسترسی به دروس تخصصی سایر گرایش­ها را دارد که اگر بتواند از میان درس­های مجاز درسی را بدون نقض کردن محدودیت ها­ی موجود در طول مسیر بیابد می­تواند آنرا اخذ کند. برای مثال اگر دانشجو درس مدار­های الکترونیکی را نگذرانده باشد و مجاز به اخذ این درس هم باشد(مثلا پیشنیازی همنیازی برقرار باشد)، می­تواند این درس را اخذ کند چرا که هیچ محدودیتی را در طول مسیر نقض نمی­کند؛ حالا فرض کنید این دانشجو درس آزمایشگاه الکترونیک دیجیتال را نگذرانده باشد و مجاز به اخذ آن نیز باشد، دراینصورت دانشجو نمی­تواند این درس را اخذ کند چرا که پیشتر گفته بودیم دانشجو دو واحد آزمایشگاه از دروس اختیاری خود را گذرانده است و اگر بخواهد این درس را اخذ کند موجب نقض محدودیت در طول مسیر می­شود(برای مثال روی یال ریشه به دروس اختیاری محدودیت حداقل و حداکثر دو واحد آزمایشگاه قرار داده شده است).

3-2-1 نقص ها و چالش­ها

ما در پیاده­سازی خود از همین ساختار برای توصیف برنامه درسی مصوب وزارت علوم استفاده کرده­ایم اما این ساختار در حالت کلی ضعف­هایی دارد و همچنین با چالش­هایی مواجه است.در ادامه به تشریح این موارد می­پردازیم.

در برنامه درسی مصوب سال 92 وزارت علوم برای رشته کامپیوتر، ایراداتی وجود داشت مثلا در جایی نام درس، گسسته و در جایی دیگر از نام ساختمان گسسته استفاده شده است در حالیکه منظور از هر دو همان گسسته بود. هنگامی که من این برنامه را بررسی می­کردم مشاهده کردم پیشنیاز­ها یا همنیاز­های یک درس در جاهای مختلف، متفاوت است برای مثال، در این برنامه درسی برای درس پایگاه داده­ها در گروه تخصصی نرم­افزار و همچنین رایانش امن، درس ساختمان­های داده به عنوان پیشنیاز در نظر گرفته شده است در حالیکه برای همین درس در گروه تخصصی فناوری اطلاعات، درس تحلیل و طراحی سیستم­ها به عنوان پیشنیاز ذکر شده است. همانطور که گفتیم این برنامه دارای ایراداتی است به همین خاطر، هنگام تحلیل این قسمت گمان کردم که این هم یک اشتباه از طرف وزارت علوم است و مدلسازی خود را با همین فرض انجام دادم اما در حقیقت چنین موردی یک اشتباه از طرف وزارت علوم نبود؛ یعنی پیشنیازی و همنیازی یک درس از خصوصیات آن درس نمی­باشد( که این همان کاری است که ما در تحلیل و پیاده­سازی خود کردیم) بلکه پیشنیازی و همنیازی، در رابطه یک درس با یک گروه قابل تعریف است. بنابراین این یکی از ضعف­های تحلیل و در پی آن پیاده­سازی ما می­باشد که برخی خصایص را بجای آنکه به رابطه بین درس و گروه اختصاص دهیم به اشتباه آنها را به عنوان خصایص خود درس در نظر گرفتیم. البته ما برای آنکه برنامه حتی در مواجه با چنین موردی به مشکل نخورد راهکاری ارائه کردیم که در آینده درباره آن توضیح خواهیم داد اما به هر حال این یک ضعف مهم در تحلیل و سپس پیاده­سازی ما به حساب می­آید.

حال به یک چالش بسیار مهم می­پردازیم، در ساختار فعلی ممکن است با یک کارت اعتباری مشخص بتوان به بعضی از درس­ها از طریق چند گروه دسترسی پیدا کرد برای مثال یک دانشجوی کامپیوتر مرتبط با برنامه درسی مصوب سال 92 وزارت علوم با گرایش مهندسی نرم­افزار و تمرکز الگوریتم و محاسبات را در نظر بگیرید، این دانشجو از سه گروه درس­های تخصصی نرم­افزار، فناوری اطلاعات و رایانش امن به درس پایگاه داده­ها دسترسی دارد، در اینگونه موارد باید مشکل را با **استنتاج منطقی** حل کرد. برای نمونه در مواجه با مثالی که بیان شد، می­توانیم اینگونه استنتاج کنیم که دانشجو باید حداقل و حداکثر 19 واحد از درس­های تخصصی گرایش نرم­افزار اخذ کند. اگر پایگاه داده ­را از این گروه اخذ نکند(یعنی اگر این درس را از یکی از دو گروه دیگر اخذ کند) محدودیت­های این گروه دسترسی پذیر ارضا نخواهد شد، پس دانشجو باید این درس را از این گروه اخذ کند و نه گروه دیگری. اما مسئله همیشه به این راحتی نیست. در مثال قبل هم استنتاج ساده بود و هم نتیجه آن به یک امر اجباری منتهی شد یعنی دانشجو باید درس پایگاه داده را از گروه خاصی اخذ می­کرد، اما می­توان برنامه­های درسی­ای را متصور شد که استنتاج در آنها به نتیجه اختیاری (نه اجبار دانشجو) منتهی شود که در این صورت باید بدانیم یک درس را از چه مسیری اخذ می­کنیم به علاوه اینکه استنتاج ممکن است بسیار پیچیده­تر باشد مثلا با اخذ یک درس بدون اینکه هیچ محدودیتی در طول مسیر نقض شود دانشجو در شرایطی قرار بگیرد که امکان فارغ­التحصیلی را از دست بدهد. با این اوصاف اگر بخواهیم مسئله را با همین مدلسازی برای حالت عمومی حل کنیم دست کم باید به پروژه خود یک **عامل استنتاج کننده** اضافه کنیم. اضافه کردن یک عامل استنتاج کننده برای من در این مرحله از نظر فنی کار آسانی نبود بعلاوه اینکه نمی­دانستم، مسئله چقدر قرار است پیچیده­تر شود. بنابراین راهکار دیگری برای مواجه با این چالش انتخاب کردم.

بجای افزودن عامل استنتاج کننده، به هنگام تعیین شدن کارت اعتباری دانشجو، فرآیندی بنام **اصلاح ساختار برنامه درسی** را معرفی کردم که در آن پس از تعیین شدن کارت اعتباری دانشجو، ساختار برنامه درسی با توجه به استنتاج­های از قبل انجام شده توسط طراح، خود را به گونه­ای تغییر می­دهد که دانشجو به هر درس، حداکثر یک مسیر قابل دسترس داشته باشد. برای نمونه دانشجوی مثال قبل را با همان کارت اعتباری در نظر بگیرید، وقتی ساختار، کارت اعتباری دانشجو را دریافت کرد اتصال بین درس پایگاه داده­ها و گروه­های فناوری اطلاعات و رایانش امن را حذف می­کند.

راهکاری که ارائه شد موجب می­شود دامنه برنامه­هایی که پروژه ما قادر به پشتیبانی از آنها است کاهش پیدا کند و در حالت کلی راه­کار مطلوبی نیست با این حال باید در نظر داشت که برنامه­های درسی فعلی وزارت علوم هم چندان برنامه‌های پیچیده­ای نیستند و همین راه­کار می­تواند در حال حاضر پاسخگوی بسیاری از نیاز­های ما باشد.

3-3 الگوریتم گام کاستن 1

در فصل قبل با گام کاستن 1 آشنا شدیم. حالا بعد از آشنا شدن با ساختار توصیف کننده برنامه درسی مصوب وزارت علوم می­خواهیم الگوریتم گام کاستن 1 را تشریح کنیم.

در گام کاستن 1 ما به عنوان ورودی با یک ساختار برنامه درسی مصوب وزارت علوم که با توجه به جدول تطبیق دروس و با استفاده از تاریخچه دانشجو پر شده است مواجهیم و همچنین اطلاعات کارت اعتباری دانشجو و یک سری اطلاعات جانبی نیز در اختیار داریم.در این الگوریتم باید به ازای هر درس موجود در ساختار، بررسی کنیم که آیا پس از اخذ شدن آن درس، آیا مسیری دسترس پذیر از درس در حال بررسی به ریشه به گونه­ای که در طول مسیر محدودیتی نقض نشود وجود دارد یا خیر؟ فقط در صورتیکه پاسخ مثبت باشد، درس مورد بررسی را در لیست نجات ذخیره می­کنیم. پس از اتمام بررسی برای همه درس­ها، درس­های موجود در لیست نجات را بعنوان خروجی الگوریتم برمی­گردانیم.

List<Course> Reduce1(Curriculum curriculum,CreditCard studCredit,

int cntPassedUnits, int currentTermNumber)

{

bool[] visited = new bool[curriculum.Courses.Count];

bool[] achivable = new bool[visited.Length];

List<Course> lst = new List<Course>();

curriculum.Courses.ForEach(course =>

{

if (Acheivable(visited, achivable, curriculum,studCredit,

course, cntPassedUnits, currentTermNumber))

{

lst.Add(course);

}

});

return lst;

}

bool Acheivable(bool[] visited, bool[] achivable,

Curriculum curriculum,CreditCard studCredit,

Course c, int cntPassedUnits,

int currentTermNumber){

//memoization :=> dynamic programming

if (visited[c.Id])

{

return achivable[c.Id];

}

if (!c.IsAvailable())

{

return false;

}

else if (c.IsPassed)

{

return false;

}

else if (c.MinRequireTerm > currentTermNumber)

{

return false;

}

else if (c.MinReuireUnits > cntPassedUnits)

{

return false;

}

for (int i = 0; i < c.PrerequisiteCourses.Count; i++)

{

var pr = c.PrerequisiteCourses[i];

if (!pr.IsPassed)

{

if (pr.NumberOfFailed > 1)

{

if (!Acheivable(visited, achivable, curriculum,

studCredit, pr, cntPassedUnits,

currentTermNumber))

return false;

}

else

{

return false;

}

}

}

for (int i = 0; i < c.RequisiteCourses.Count; i++)

{

var r = c.RequisiteCourses[i];

if (!r.IsAvailable())

return false;

if (r.IsPassed == false && !Acheivable(visited, achivable,

curriculum, studCredit, r, cntPassedUnits,

currentTermNumber))

return false;

}

var res = if exists a path from c to root then return true

else return false:inputs(curriculum,studCredit,c);

//memoization

visited[c.Id] = true;

achivable[c.Id] = res;

return res;

}

3-4 الگوریتم مرحله 1 از گام کاستن 2

در گام کاستن 1 تعدادی از درس­ها **نجات پیدا کردند** اما ممکن است همه­ی این درس­های نجات پیدا کرده در ترم جاری توسط دانشگاه ارائه نشده باشند. در این مرحله لیستی از دورس نجات پیدا کرده از مرحله قبل را به همراه برنامه دروس ارائه شده توسط دانشگاه در ترم جاری و به همراه اطلاعات جانبی به الگوریتم ارسال می­کنیم. سپس الگوریتم سطرهایی از سطر­های ارائه شده از جدول برنامه دروس ارائه شده توسط دانشگاه را که درس­ متناظر آن­ در لیست دروس نجات یافته از مرحله قبل موجود باشد، برای این مرحله در لیست نجات مخصوص خود ذخیره می­کند و در پایان، الگوریتم همه سطر­های نجات پیدا کرده را برمی­گرداند.

توجه به این نکته نیز ممکن است خالی از لطف نباشد که ما در کد­ها یا شبه­کدهایی که می­آوریم هم برای مفهوم لیست و هم برای مفهوم مجموعه از نمایش List<T> کمک می­گیریم. در واقع مجموعه را لیستی با عناصر متمایز در نظر می­گیریم که تعریف درستی نیست اما همین برای کار ما کفایت می­کند.

List<OfferedCoursesRow> Reduce2(List<Course> courses, List<OfferedCoursesRow> offeredCoursesRows, Gender studGender

,bool capacityFiltering=false)

{

var groups = from r in offeredCoursesRows

where ((r.Gender == studGender

|| r.Gender == Gender.COEDUCATIONAL) &&

(capacityFiltering?CalculateCapacity(r):true))

group r by r.CodeInDesUni;

var newLst = new List<OfferedCoursesRow>();

foreach (var group in groups)

{

var q = courses.FirstOrDefault(c => c.CodeInDesUni == group.Key);

if (q != null)

{

foreach (var row in group)

{

newLst.Add(row);

}

}

}

return newLst;

}

4-5 ساختار­ها و الگوریتم­های اصلی مرحله2 از گام کاستن2

بعد از مرحله1 از گام کاستن2 ما با تعدادی سطر از جدول دروس ارائه شده در ترم توسط دانشگاه مواجه­ایم که هنوز احتمال انتخاب شدن برای آنها وجود دارد. در فصل قبل گفتیم در این مرحله می­خواهیم با دریافت اولویت­های کاربر سعی کنیم باز­هم فضای مسئله را کاهش دهیم. حال در اینجا باید تصمیم بگیریم که چگونه می­توانیم اولویت­های فهرست شده را از کاربر دریافت کنیم و چگونه این کاهش را باید صورت داد؟! در ادامه به این موضوعات می­پردازیم.

4-5-1 درس­ها و سطر­های رنگی

برای آنکه کاربر را قادر سازیم تا بتواند الویت­های انتخاب شده در فصل قبل را بیان کند سه رنگ سبز، سفید و قرمز را در نظر می­گیریم که هر درس و همچنین هر سطر می­تواند یکی از این سه رنگ را داشته باشد. رنگ سبز برای یک سطر، به معنای این است که کاربر می­خواهد آن را حتمی اخذ نماید. رنگ سفید برای یک سطر به معنای اینست که کاربر در رابطه با اخذ کردن یا اخذ نکردن آن سطر نظر قاطعی ندارد و رنگ قرمز به معنای اینست که کاربر به هیچ وجه نمی­خواهد سطر فعلی را اخذ نماید. به طور مشابه درس سبز به معنای این است که کاربر می­خواهد درس فعلی حتما در برنامه ترم جاری­اش وجود داشته باشد، درس سفید یعنی این که در این رابطه نظر قاطعی ندارد و درس قرمز هم یعنی این درس به هیچ عنوان در ترم جاری اخذ نشود. با استفاده از مفهوم درس­ها و سطر­های رنگی می­توان همه اولویت­های انتخاب شده فصل قبل را توصیف کرد.

4-5-2 الگوریتم های مرحله2 از گام کاستن2

در حین توصیف اولویت­ها توسط کاربر ما باید، بررسی کنیم که اولویت­های کاربر مجاز بوده و با هم در تناقص نباشند.در آخر پس از اتمام کار توصیف اولویت­های کاربر نیز باید ورودی­ها را برای الگوریتم مهیا کنیم. یکی از مهمترین بخش­های این اعتبارسنجی این است که مجموعه درس­های سبز انتخاب شده توسط کاربر باید ساختار را در وضعیت معتبر نگه دارد به علاوه اینکه باید حداقل یک برنامه ترمی مجاز و ممکن برای این مجموعه وجود داشته باشد. در این بخش به این موضوع که چطور می­توانیم متوجه شویم پس از سبز کردن تعدادی درس آیا در وضعیت معتبر قرار داریم یا نه می­پردازیم. توضیح اینکه چطور می­توانیم متوجه شویم برای یک مجموعه درس، برنامه ترمی مجاز و ممکنی وجود دارد یا نه را تا رسیدن به بخش مربوط به الگوریتم پیشنهاد برنامه ترمی به تاخیر می­اندازیم.

ما به عنوان ورودی، ساختار برنامه ترمی­ای که با توجه به جدول تطبیق دروس و تاریخچه دانشجو پر شده است را به علاوه کارت اعتباری دانشجو و مجموعه دروس سبز رنگ درخواستی دانشجو دریافت می­کنیم. برای آنکه بتوانیم تشخیص دهیم که آیا با اخذ کردن مجموعه دروس مشخص شده، ساختار در وضعیت معتبری است یا خیر کافی است بدانیم پس از اخذ همه این دروس، قانونی نقض شده است یا نه. اگر قانونی نقض نشده باشد یعنی در وضعیت معتبری قرار داریم و در واقع مجاز به اخذ آن مجموعه دروس می­باشیم. اما در غیرآن صورت در وضعیت نامعتبر بوده و مجاز به اخذ آن مجموعه از دروس نمی­باشیم.اما پیش از آنکه راهکاری برای این اعتبارسنجی پیشنهاد کنیم باید قوانین را بشناسیم و بدانیم چگونه باید آنها را بررسی کنیم.

قوانین را می­توان از زوایای مختلف به شکل­های گوناگون تقسیم بندی کرد ما در اینجا آنها را به دو نوع **قوانین پایه** و **قوانین ثانویه** تقسیم­بندی کردیم. قوانین پایه، قوانینی هستند که نسبت به برنامه درسی مصوب وزارت علوم ثابت اند مثلا چون در برنامه درسی مصوب سال 92 وزارت علوم برای رشته مهندسی کامپیوتر، مفهومی مثل پیشنیازی و همینازی دروس وجود دارد و از آنجا که در این برنامه ذکر شده که: "این برنامه از تاریخ تصویب به مدت پنج سال قابل اجراء است و پس از آن نیازمند بازنگری است." ما انتظار داریم پیشنیازی و همنیازی و همچنین بصورت کلی قوانین مربوط به این برنامه در طول پنج سال عمر آن رعایت شده و ثابت بمانند. اما در کنار این قوانین، مواردی هستند که در طول عمر برنامه درسی مصوب وزارت علوم ممکن است تغییر کنند(برای مثالی از این قانون، مجاز بودن اخذ حداکثر یک درس معارف در یک ترم تحصیلی را در نظر بگیرید). پس ما نیاز است این دو نوع از قوانین را برای تشخیص معتبر بودن یا نبودن وضعیت پس از انتخاب درس­های سبز بررسی کنیم. بررسی کردن قوانین ثانویه معمولا سریع­تر انجام می­شود از این جهت در بررسی­های خود ترجیح دادیم ابتدا این قوانین را بررسی کنیم.

bool IsValidState(Curriculum curriculum,CreditCard studCredit, List<int> takenCoursesId)

{

//secondary rules

bool o = SecondaryRulesStateValidator.IsValidState(curriculum,

studCredit, takenCoursesId);

if (!o) return false;

//basic rules

return BasicRulesStateValidator.IsValidState(curriculum,studCredit,

takenCoursesId);

}

یکی از قوانینی که در قسمت اعتبارسنجی قوانین پایه باید صورت پذیرد، بررسی این است که آیا پس از اخذ مجموعه دروس سبز ورودی، محدودیت­ قابل دسترسی نقض می­شود یا نه. برای بررسی این مورد شبه کدی ارائه نمی­کنیم درعوض عملکرد کلی آن را شرح می­دهیم. برای تشخیص این موضوع،ابتدا با استفاده از دروس سبز ورودی باید اطلاعات در ارتباط با ساختار توصیف کننده برنامه درسی مصوب وزارت علوم را بروز رسانی کرد سپس باید از ریشه شروع به جستجو کرده و از همه گروه­های دسترسی پذیر عبور کرد، چنانچه در مسیر جستجو موردی مشاهده شد که دسترسی به آن مجاز بود اما محدودیتی نیز به واسطه ان دسترسی نقض می­شود، یعنی وضعیت ساختار برنامه درسی نامعتبر است و اگر تا پایان جستجو چنین موردی یافت نشد یعنی در وضعیت معتبر قرار دارد.

4-6 الگوریتم پیشنهاد برنامه­های ترمی

در این بخش می­خواهیم شما را با الگوریتم اصلی برنامه که همان الگوریتم پیشنهاد برنامه(های) ترمی است آشنا کنیم. اما از آنجا که خود الگوریتم از زیرساخت­ها و حل مسئله­های دیگر بهره می­برد ما ابتدا به این موضوعات پرداخته سپس به خود الگوریتم می­پردازیم.

4-6-1 زیرساخت اول

ما پیشتر در این فصل هنگام مواجه شدن با مسئله­ی اعتبارسنجی درس­های سبز، گفتیم که درباره آن بعدا صحبت می­کنیم؛ اکنون زمان آن رسیده است. جدول برنامه درسی ارائه شده در ترم را که توسط دانشگاه ارائه می­شود به یاد بیاورید. گفته بودیم که؛ این جدول شامل سطر­هایی است که هر سطر آن نیز شامل ستون­هایی است. به علاوه برخی از ستون­ها را نام برده بودیم. یک ستون از ستون­های هر سطر، مربوط به شماره و گروه درس(که از آن می­توان کد درس را نیز بدست آورد) می­باشد.

حال فرض کنید یک مجموعه از درس­های سبز را بعنوان ورودی به شما می­دهند و می­پرسند با فرض این که اخذ این دروس، سایر قوانین را نقض نکرده باشد، همه برنامه­های ترمی مجاز و ممکن را برای آن­ها بعنوان خروجی برگردان.

حال فرض کنید ما با پیش پردازش از یک نوع مدلسازی خاص استفاده کرده­باشیم که در آن هر درس به همه سطر‌های موجود در ترم آن دسترسی دارد(همه سطر­های جدول برنامه ترمی ارائه شده در ترم توسط دانشگاه که مربوط به آن درس است) و هر سطر هم به درسی که به آن ارجاع می­کند، ارجاع دارد.

class OfferedCourse{

Course Course;

List<OfferedCourseRow> OfferedCourseRows;

Color Color;

}

class OfferedCourseRow{

OfferedCourse OfferedCourse;

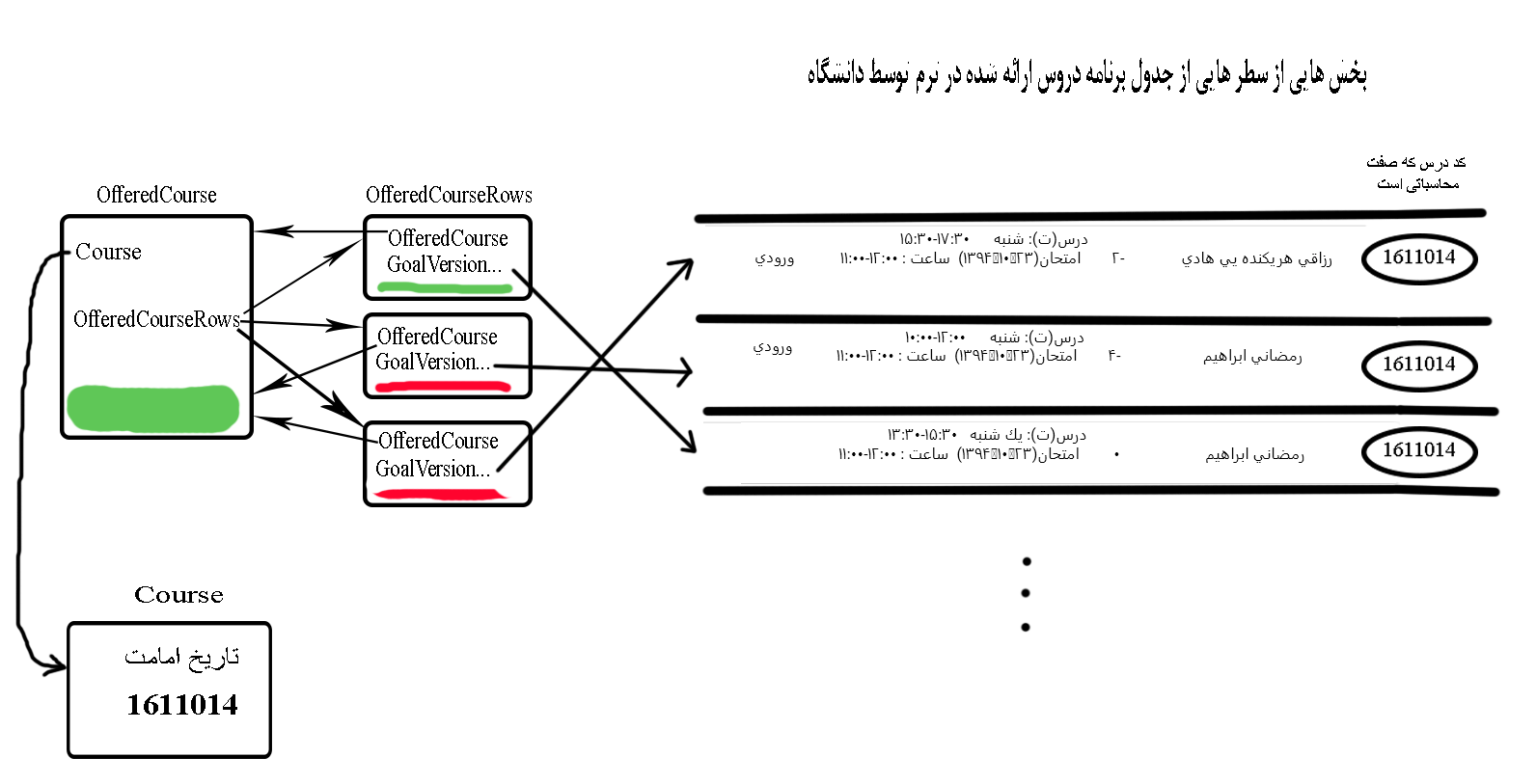
Main.OfferedCoursesRow GoalVersionOfferedCourseRow;

Color Color;

}

در این مدلسازی OfferedCourse نماینده یک درس می­باشد و دارای فیلد­های Course،OfferedCourseRows و Color است. Course نماینده یک درس می­باشد(درسی مشابه آنچه از ساختار توصیف کننده برنامه درسی مصوب وزارت علوم می­شناسیم)، OfferedCourseRows به سطر­هایی ارجاع دارد که GoalVersionOfferedCourseRow آنها به سطری از جدول برنامه دروس ارائه شده در ترم توسط دانشگاه ارجاع داشته باشد که کد درس آن با کد درس Course در OfferedCourse یکسان باشد و Color همان سه رنگ سبز، سفید و قرمز می­تواند باشد.

OfferedCourseRow نماینده یک سطر است که شامل OfferedCourse،GoalVersionOfferedCourseRow و Color می­شود. خصوصیت OfferedCourse به نمونه ای از کلاس OfferedCourse که به این سطر ارجاع دارد اشاره می­کند. GoalVersionOfferedCourseRow از نوع Main.OfferedCourseRow تعریف شده است. منظور ما از این نوع همان OfferedCourseRow ای است که در شبه­کدهای بخش ­های قبلی استفاده کردیم که نماینده یک سطر از جدول دروس ارائه شده در ترم توسط دانشگاه بود و بخاطر اینکه بتوانیم آنرا از نوع OfferedCourseRow ای که در اینجا تعریف کردیم تفکیک کنیم از این نوع نمایش استفاده شده است. Color نیز مانند قبل می­تواند یکی از سه مقدار سبز، سفید و قرمز را داشته باشد.

**

شکل 2-3 مدل گرافیکی نمونه­ای از ارتباطات Course،OfferedCourse،OfferedCourseRow و Main.OfferedCourseRow

علاوه بر این ساختار­ها، دو ساختار دیگر نیز معرفی می­کنیم که عبارتند از Box و Row و بصورت زیر قابل توصیف هستند:

class Row{

List<OfferedCourseRow> Columns;

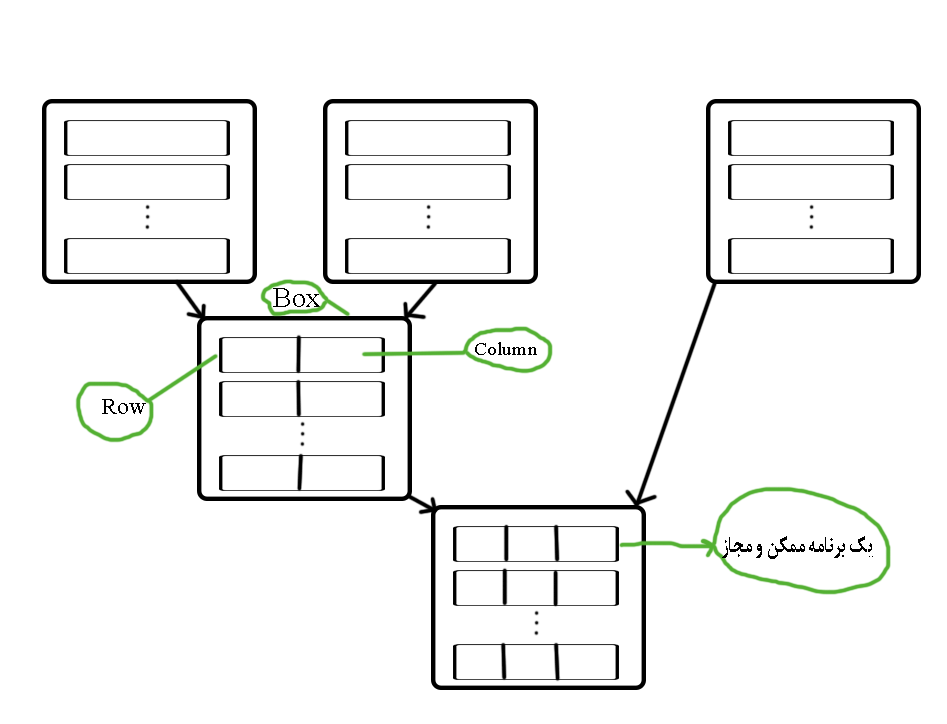
}

class Box{

List<Row> Rows;

}

ما به ازای هر درس از مجموعه دروس ورودی یک Box می­سازیم و سپس به صورت درخت مانند هر مرحله، دو به دو همه برنامه­های ممکن و مجاز، یعنی آن Boxهایی که Columnهای آنها با هم تداخل زمانی ندارند را محاسبه می­کنیم. این کار را تا بررسی آخرین Box با تعداد Column برابر یک ادامه می­دهیم. اگر در طول محاسبات مرحله­ای، به Box ای برخوردیم که طول Rows آن پس از انجام محاسبه صفر بود یعنی برنامه ممکن و مجازی برای این مجموعه دروس وجود ندارد و اگر هرگز به چنین Boxای برنخوردیم، آخرین Boxای که محاسبه می­کنیم شامل تمام برنامه­های مجاز و ممکن خواهد بود.



شکل 3-3 شیوه محاسبه کردن تمام برنامه­های ممکن و مجاز برای یک مجموعه درس ورودی سه تایی

Box CreateBoxForOfferedCourse(OfferedCourse offeredCourse)

{

Box box = new Box();

offeredCourse.OfferedCourseRows.ForEach(r =>

{

if (r.Color != RED)

{

Row row = new Row();

row.Columns.Add(r);

box.Rows.Add(row);

}

});

return box;

}

تابع CreateBoxForOfferedCourse یکی از دروس موجود در مجموعه دروس ورودی را بعنوان آرگومان گرفته Box متناظر با آن را ایجاد کرده، برمی­گرداند.

List<Box> Validate(List<Box> boxes, bool examCollideCheck = false)

{

if (boxes.Count == 1 && boxes[0].Rows.Count == 0)

return null;

List<Box> bxs = new List<Box>();

bxs.Add(boxes[0]);

bool con0 = true;

for (int i = 1; i < boxes.Count; i++)

{

var b1 = bxs[0];

var b2 = boxes[i];

Box newBox = new Box();

for (int j = 0; j < b2.Rows.Count; j++)

{

for (int k = 0; k < b1.Rows.Count; k++)

{

bool con = true;

Row newRow = new Row();

for (int m = 0; m < b1.Rows[k].Columns.Count; m++)

{

bool collide = DoTheyCollide(b1.Rows[k].Columns[m]

.GoalVersionOfferedCourseRow

.TimeAndSitesAndExam

.TimeAndSites,

b2.Rows[j].Columns[0]

.GoalVersionOfferedCourseRow

.TimeAndSitesAndExam

.TimeAndSites);

if (collide){

con = false;

break;

}

if (examCollideCheck &&

DoTheyExamCollide(b1.Rows[k].Columns[m]

.GoalVersionOfferedCourseRow

.TimeAndSitesAndExam.Exam,

b2.Rows[j].Columns[0]

.GoalVersionOfferedCourseRow

.TimeAndSitesAndExam.Exam)){

con = false;

break;

}

newRow.Columns.Add(b1.Rows[k].Columns[m]);

}

newRow.Columns.Add(b2.Rows[j].Columns[0]);

if (con) newBox.Rows.Add(newRow);

}

}

if (newBox.Rows.Count == 0){

con0 = false;

break;

}

else bxs[0] = newBox;

}

if (con0) return bxs;

else return null;

}

تابع Validate بعنوان آرگومان یک لیست از Boxها را گرفته، با بررسی دو به دو آنها، در آخر اگر برنامه(های) ممکن و مجازی وجود داشته باشد، همه برنامه­های ممکن و مجاز را محاسبه کرده و برمی­گرداند و درغیراینصورت null خروجی تابع خواهد بود.

Validate(List<OfferedCourse> offeredCoursesList, bool examCollideCheck = false)

{

List<Box> lst = new List<Box>();

for (int i = 0; i < offeredCoursesList.Count; i++)

lst.Add(CreateBoxForOfferedCourse(offeredCoursesList[i]));

return Validate(lst, examCollideCheck);

}

این تابع Validate، مجموعه دروسی که می­خواستیم از اول این بحث، برنامه­های ممکن و مجاز را برایشان محاسبه کنیم بعنوان ورودی گرفته Boxهای متناظر آنها را ایجاد کرده در لیستی ذخیره می­کند سپس با فراخوانی تابع Validateای که قبل تر بررسی­اش کردیم که لیستی از Boxها را بعنوان ورودی می­گیرد، همه برنامه­های ممکن و مجاز را در صورتیکه حداقل یک برنامه ممکن و مجاز وجود داشته باشد بر­می­گرداند و چنانچه چنین برنامه­ای وجود نداشته باشد null بعنوان خروجی تابع در نظر گرفته می­شود.

4-6-2 بخش اصلی الگوریتم

الگوریتم پیشنهاد برنامه(های) ترمی بعنوان ورودی به ترتیب لیستی از امتیاز دروس،لیستی از درس­های سفید، لیستی از درس­های سبز، حداقل واحد مجاز برای برنامه ترمی، حداکثر واحد مجاز برای برنامه ترمی، مجموع تعداد واحد­های درس­های سبز، مجموع امتیازات دروس سفید، ساختار توصیف کننده برنامه ترمی مصوب وزارت علوم که با اطلاعات مناسب پر شده است، کارت اعتباری دانشجو، همه برنامه­های ممکن و مجاز برای دروس سبز، حالت بررسی تصادم امتحانات و تایم اجرا الگوریتم را بعنوان ورودی دریافت می­کند.

الگوریتم ابتدا از میان درس­های سفید ورودی، بصورت تصادفی با احتمال **امتیاز درس ضرب در صد تقسیم بر مجموع امتیازات دروس سفید** درس­هایی را جهت اخذ شدن انتخاب می­کند. اگر مجموع واحد­های دروس انتخاب شده در این مرحله به علاوه مجموع واحد­های دروس سبز، در محدوده مجاز حداقل و حداکثر واحد مجاز یک برنامه ترمی قرار نداشت این مرحله تکرار می­شود. در غیر اینصورت الگوریتم چک می­کند آیا با اخذ مجموعه دروسی شامل درس­های سبز و درس­های سفیدِ انتخابیِ فعلی، آیا ساختار در وضعیت معتبری قرار دارد یا خیر؟ اگر پاسخ منفی باشد الگوریتم مجدادً باید دنبال مجموعه دروس سفید جدیدی برای بررسی بگردد اما اگر پاسخ مثبت بود الگوریتم همه برنامه­های ممکن و مجاز از مجموعه شامل دروس سبز و سفید انتخابی فعلی را در صورتیکه وجود داشته باشند، می­باید. پس از آن هر کدام از این برنامه­ها به نمونه­ای بنام \_ChoosedWeeklyProgramManager، درخواستی برای قرار گرفتن در لیست نهایی می­دهند. \_ChoosedWeeklyProgramManager وظیفه دارد برای هر برنامه­درخواستی ارزشی براساس تابع ارزیابی در نظر بگیرد. به علاوه وظیفه دیگر آن گزینش تعداد مشخصی از بهترین برنامه­های غیر تکراری یافته شده است. الگوریتم همه این تکرار ها را در مدت زمان مشخصی انجام می­دهد و پس از آن، تعداد مشخصی از بهترین برنامه ترمی‌هایی را که یافته باز می­گرداند.

Algo(double[] courseScore, List<OfferedCourse> whiteCourses, List<OfferedCourse> greenCourses, int minUnits, int maxUnits, int greenCoursesUnits, double whiteCoursesTotalScore,Curriculum curriculum, CreditCard studCredit,List<Box> GreenCoursesBoxes, bool examCollideCheck, int timeout)

{

var \_ChoosedWeeklyProgramManager= new AlgorithmTopWeeklyProgramManager();

Execute.For(timeout){

Random random = new Random();

int CurrentSelectedUnits=0;

bool[] Selected;

List<int> takenCoursesId = new List<int>();

while (true){

while (true){

do{

Selected = new bool[whiteCourses.Count];

CurrentSelectedUnits = 0;

takenCoursesId.Clear();

//init and assigning values to Selected array

for (int i = 0; i < Selected.Length; i++){

var offeredCourse = whiteCourses[i];

var course = whiteCourses[i].Course;

var pc = random.Next(100);

if (CurrentSelectedUnits + course.Units + greenCoursesUnits >

maxUnits) continue;

if (pc <= courseScore[course.Id] \* 100.0 /

whiteCoursesTotalScore){

Selected[i] = true;

CurrentSelectedUnits += course.Units;

takenCoursesId.Add(course.Id);

}

}

}while (minUnits > CurrentSelectedUnits + greenCoursesUnits);

greenCourses.ForEach(gc => takenCoursesId.Insert(0, gc.Course.Id));

//two step validation

bool o = IsValidState(curriculum,studCredit,takenCoursesId);

if (o) break;

}

List<Box> boxes = new List<Box>();

GreenCoursesBoxes.ForEach(b => boxes.Add(b));

for (int i = 0; i < Selected.Length; i++){

if (Selected[i]){

Box b = CreateBoxForOfferedCourse(whiteCourses[i]);

boxes.Add(b);

}

}

List<Box> res = Validate(boxes, examCollideCheck);

if (res != null) \_ChoosedWeeklyProgramManager.TryAddNewWeeklyProgram(res[0].Rows);

}

}

return \_ChoosedWeeklyProgramManager;

}

همانطور که دیدیم الگوریتم بالا ورودی­های خاصی دارد که به نظر نمی­رسد مهیا کردن همه آنها توسط مشتری(کسی که از الگوریتم استفاده می­کند یا آن را فراخوانی می­کند) بصورت مستقیم کار معقولی باشد. در واقع نیز قرار نیست همه این ورودی ها را مشتری تهیه کند بلکه وظیفه مشتری فقط تهیه ورودی­های ضروری است، در این قسمت ما شبه­کد دیگری ارائه می­کنیم که در آن اطلاعات مورد نیاز برای الگوریتم، با توجه به ورودی­های ضروری مهیا شده از طرف مشتری تولید می­شود و سپس با فراخوانی آنچه در شبه­کد قبلی دیدیم در یک باز­ه زمانی مشخص، سعی می­کنیم تعدادی برنامه ترمی مناسب به کاربر پیشنهاد دهیم.

MainAlgo(List<OfferedCourse> inputs, Curriculum curriculum

,CreditCard studCredit, int minUnits, int maxUnits

, bool examCollideCheck, int timeout){

double[] courseScore = new double[curriculum.Courses.Count];

double numberOfGreenUnits = 0;

List<OfferedCourse> greenCourses = new List<OfferedCourse>();

//white courses that have at least one white row

List<OfferedCourse> whiteCourses = new List<OfferedCourse>();

int greenUnits = 0;

for (int i = 0; i < inputs.Count; i++){

var offeredCourse = inputs[i];

if (offeredCourse.Color == Green){

greenCourses.Add(offeredCourse);

greenUnits += offeredCourse.Course.Units;

}

else if (offeredCourse.Color == WHITE){

for (int r = 0; r < offeredCourse.OfferedCourseRows.Count; r++){

var offeredCourseRow = offeredCourse.OfferedCourseRows[r];

if (offeredCourseRow.Color == WHITE){

whiteCourses.Add(offeredCourse);

break;

}

}

}

var course = offeredCourse.Course;

courseScore[course.Id] += (course.Units \* 2 - 1);

for (int c = 0; c < course.PrerequisiteCourses.Count; c++){

var preCourse = course.PrerequisiteCourses[c];

if (!preCourse.IsPassed && preCourse.IsAvailable()){

if (preCourse.NumberOfFailed > 1)

courseScore[preCourse.Id] += (course.Units \* 2 - 1) \* 0.20;

else

courseScore[preCourse.Id] += (course.Units \* 2 - 1) \* 0.25;

}

}

for (int c = 0; c < course.RequisiteCourses.Count; c++){

var reqCourse = course.RequisiteCourses[c];

if (!reqCourse.IsPassed && reqCourse.IsAvailable())

courseScore[reqCourse.Id] += (course.Units \* 2 - 1) \* 0.20;

}

}

double whiteCoursesTotalScore = 0;

whiteCourses.ForEach(i =>{whiteCoursesTotalScore +=

courseScore[i.Course.Id];});

List<Box> greenCoursesBoxes = new List<Box>();

greenCourses.ForEach(gc =>{

numberOfGreenUnits += gc.Course.Units;

Box b = CreateBoxForOfferedCourse(gc);

greenCoursesBoxes.Add(b);

});

if (greenCoursesBoxes.Count > 0){

List<Box> res = Validate(greenCoursesBoxes, examCollideCheck);

if (res != null){

greenCoursesBoxes.Clear();

greenCoursesBoxes.Add(res[0]);

}

}

return Algo(courseScore, whiteCourses, greenCourses, minUnits,

maxUnits, greenUnits, whiteCoursesTotalScore, curriculum

,studCredit, greenCoursesBoxes, examCollideCheck,timeout);

}

امتیاز دادن به درس­ها بستگی به شرایط مسئله یا نظر طراح، می­تواند متفاوت باشد. ساده­ترین راهکار این است که امتیاز همه درس­ها را یکسان در­نظر بگیریم در این صورت همه درس­ها برای انتخاب شدن در برنامه شانس برابری خواهند داشت اما بدون توضیح بیشتر این راهکار خوبی به نظر نمی­رسد.در عوض استفاده از یک **تابع اکتشافی**، جذاب­تر بنظر می­رسد. تابع اکتشافی ما سعی می­کند معیاری برای خوب بودن یک درس ارائه کرده و احتمال انتخاب شدن درس­های بهتر را بیشتر کند و این درحالی است که برای درس­های بدتر هم شانس انتخاب شدن قائل می­شود. همانطور که از شبه­کد هم قابل برداشت است ما امتیاز هر درس را به صورت زیر تعریف کردیم:

( تعداد واحد درس \* 2 - 1 ) +

( تعداد واحد­های دروس موجود پاس نشده­ای که این درس در حال حاضر پیشنیاز آنها است \* 0.25 )

+ ( تعداد واحد­های دروس موجود پاس نشده­ای که در حال حاضر این درس همنیاز آنها است \* 0.20 )

در طراحی این تابع فرض ما این اینست که پیشنیاز­ی و همنیازی دروس قانونی پایه­ای است که برای همه برنامه­های مصوب وزارت­علوم خواه برنامه­های فعلی باشد خواه برنامه­های آینده ثابت و موجود است و به همین دلیل وابسته کردن فرمول بالا به این قانون، ایرادی ندارد. بعلاوه معیاری که ما برای امتیاز دهی در نظر گرفتیم مثل این است که بگوئیم **درسی بهتر است که تعداد واحد­های بیشتری داشته باشد و با اخذ یا پاس­کردن آن بتوان تعداد دروس بیشتری را قابل اخذ کرد**. اما همانطور که گفته بودیم این فرمول بسته به سلیقه طراح می­تواند متفاوت باشد.

تا به اینجا بخش­های مهمی از الگوریتم را تشریح کردیم اما هنوز معیاری برای ارزش­گذاری برنامه­های ترمی ارائه نکردیم. در رابطه با این موضوع نیز باید گفت معیاری که برای این ارزش­گذاری انتخاب می­شود بسته به سلیقه طراح می­تواند متفاوت باشد.

معیار اصلی­ای که ما در اینجا از آن استفاده کردیم، حداقل فاصله بین کلاس­ها است. البته این تنها معیاری نیست که در نظر گرفتیم چرا که در این صورت برنامه­هایی که تعداد واحد بیشتری دارند در مقایسه با برنامه­های با تعداد واحد کمتر تقریبا هیچ شانسی برای گزینش پیدا نمی­کنند و این درحالی است که کاربر حداقل و حداکثر واحدی را برای برنامه­ترمی پیشنهادی مشخص می­کند. بنابراین معیاری که از آن استفاده کردیم **حداقل فاصله بین کلاس­ها به ازای هر واحد** است.

CalculateWeeklyProgramValue(List<OfferedCourseRow> lst)

{

double res = 0;

int units = 0;

var rows = new SortedList<int,TimeAndSite>();

foreach (var item in lst){

if (item.GoalVersionOfferedCourseRow.TimeAndSitesAndExam

!= null && item.GoalVersionOfferedCourseRow

.TimeAndSitesAndExam.TimeAndSites != null){

foreach (var timeAndSite in

item.GoalVersionOfferedCourseRow.

TimeAndSitesAndExam.TimeAndSites){

rows.Add((((int)timeAndSite.Day) \* 60 \* 24 +

timeAndSite.StartTime.Hour \* 60

+ timeAndSite.StartTime.Minute)

, timeAndSite);

}

}

units += item.OfferedCourse.Course.Units;

}

for (int i = 1; i < rows.Count; i++){

int dt = (((int)rows.Values[i].Day) \* 60 \* 24 +

rows.Values[i].StartTime.Hour \* 60

+ rows.Values[i].StartTime.Minute) -

(((int)rows.Values[i - 1].Day) \* 60 \* 24 +

rows.Values[i - 1].FinishTime.Hour \* 60

+ rows.Values[i - 1].FinishTime.Minute);

res += dt;

}

return (res \* -1.0 \* 1.3) / units;

}

هر چه ارزش محاسبه شده برای یک برنامه ترمی کمتر منفی باشد آن برنامه بهتر است در واقع اینطور در نظر گرفتیم که بهترین حالت وقتی است که تمام وقت دانشجو در طول هفته برای خودش باشد.

3-7 ارائه ترتیبی قابل اخذ برای سطر­های یک برنامه ترمی

در فصل اول در رابطه با سیستم گلستان گفتیم ،در حال حاضر در هنگام انتخاب واحد در این سیستم شما باید واحد ها را به ترتیبی ثبت کنید که همواره برنامه ترمی شما در وضعیت معتبری قرار داشته باشد یعنی برای مثال جهت اخذ درسی که همنیازی دارد ابتدا باید همنیاز آن درس را وارد کرده سپس خود درس را وارد کنید. حالا که الگوریتم پیشنهاد برنامه ترمی را معرفی کردیم وقت آن رسیده ­بررسی کنیم چگونه می­توانیم سطر­های یک برنامه ترمی را با ترتیبی قابل اخذ برای سیستم گلستان مرتب کنیم.

راه­حل بسیار ساده است، تنها کافیست یک گراف بر مبنای پیشنیازی و همنیازی مجموعه دروس موجود در برنامه ترمی ایجاد کنید و سپس با اجرای الگوریتمی که مرتب سازی توپولوژیکی روی این گراف انجام دهد، شما به یک ترتیب قابل اخد برای آن دروس دست خواهید یافت.

3-8 خلاصه

در این فصل بحث را با معرفی ساختار توصیف کننده برنامه درسی مصوب وزارت علوم شروع کردیم. پس از معرفی این ساختار نقض­ها و چالش­هایی که ساختار معرفی شده با آنها مواجه بود را بررسی کرده و خط مشی که در این خصوص اتخاذ کردیم را بیان نمودیم. در ادامه الگوریتم­های مربوط به ایده کاستن را که در فصل قبل با آنها آشنا شدیم شرح دادیم و در طول این کار با مفاهیمی مانند درس­ها و سطر­های رنگی آشنا شدیم. بعد از این مرحله به الگوریتم اصلی پروژه خود یعنی الگوریتم پیشنهاد برنامه­های ترمی رسیدیم و پس از تشریح زیر­ساخت­های آن به تشریح خود الگوریتم پرداختیم. در آخر نیز راهکاری برای دستیابی به ترتیبی قابل اخذ برای دروس یک برنامه ترمی پیشنهاد دادیم.

فصل چهارم:

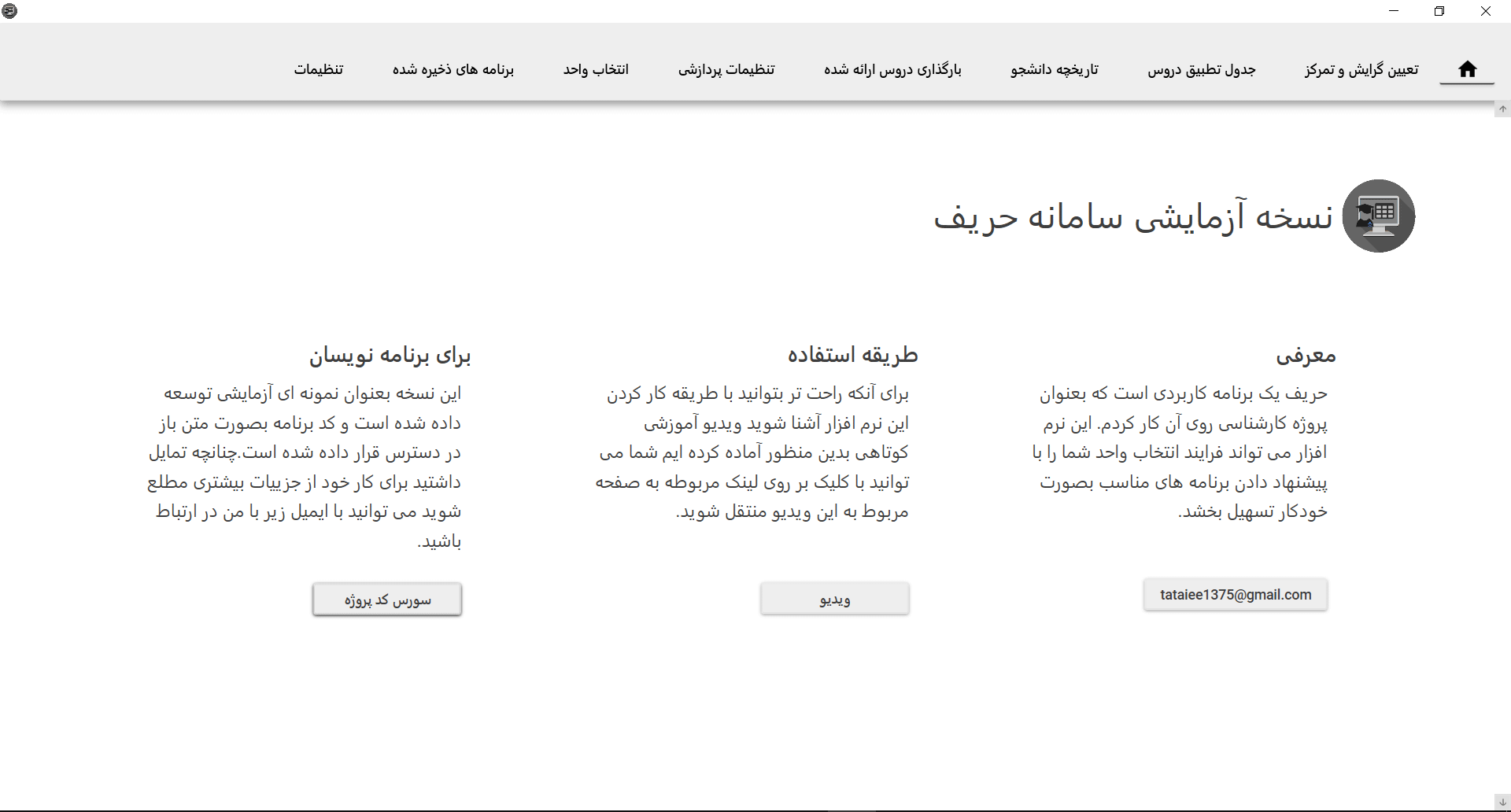
نسخه آزمایشی و نتایج آزمایشات

4-1 مقدمه

در این فصل ابتدا شما را با یک نسخه آزمایشی توسعه داده شده بر مبنای برنامه درسی مصوب سال 92 وزارت علوم برای رشته مهندسی کامپیوتر آشنا خواهیم کرد. سپس با اشاره به تعدادی از برنامه­های ترمی پیشنهادی توسط نسخه آزمایشی که آنها را در طول آشنایی با این نسخه، طرح می­کنیم و برخی از نکات و توضیحات را در رابطه با نتایج حاصل از آزمایشات بیان می­کنیم.

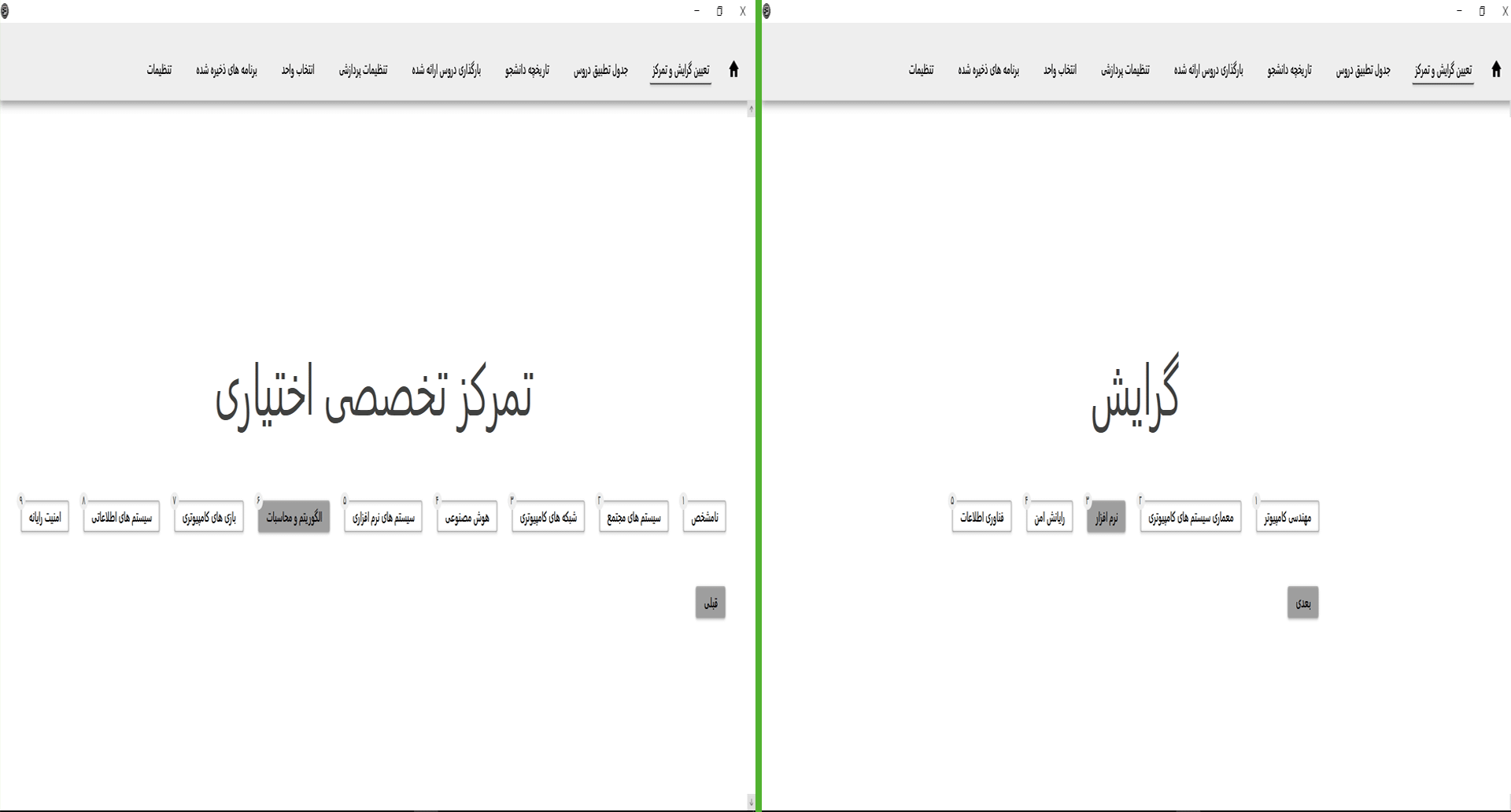
4-2 آشنایی با نسخه آزمایشی

نسخه آزمایشی، برنامه­ای است که بعنوان نمونه با توجه به آنچه در فصل های قبل بیان کردیم توسعه داده شده است. این نسخه بر مبنای برنامه درسی سال 92 مصوب وزارت علوم برای رشته مهندسی کامپیوتر با زبان سی شارپ پیاده­سازی شده است. در ادامه شما را با این نسخه آزمایشی آشنا می­کنیم.



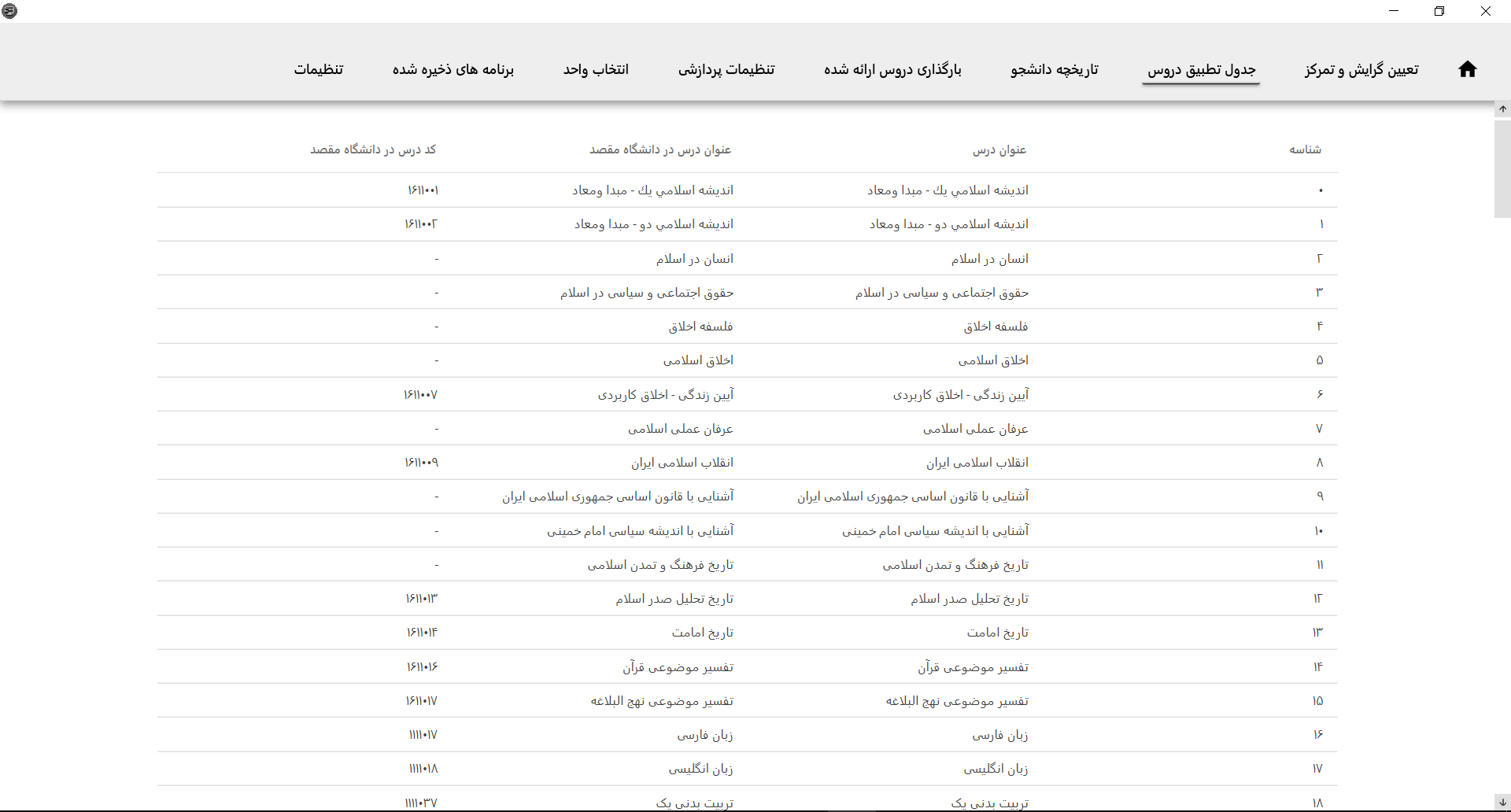
شکل 4-1. تصویری از صفحه خانه نرم­افزار

برنامه بجز صفحه خانه که صرفا جنبه نمایشی دارد شامل بخش­هایی به نام­های تعیین گرایش و تمرکز، جدول تطبیق دروس، تاریخچه دانشجو، بارگذاری دروس ارائه شده، تنظیمات پردازشی، انتخاب واحد، برنامه­های ذخیره شده و تنظیمات می­باشد.



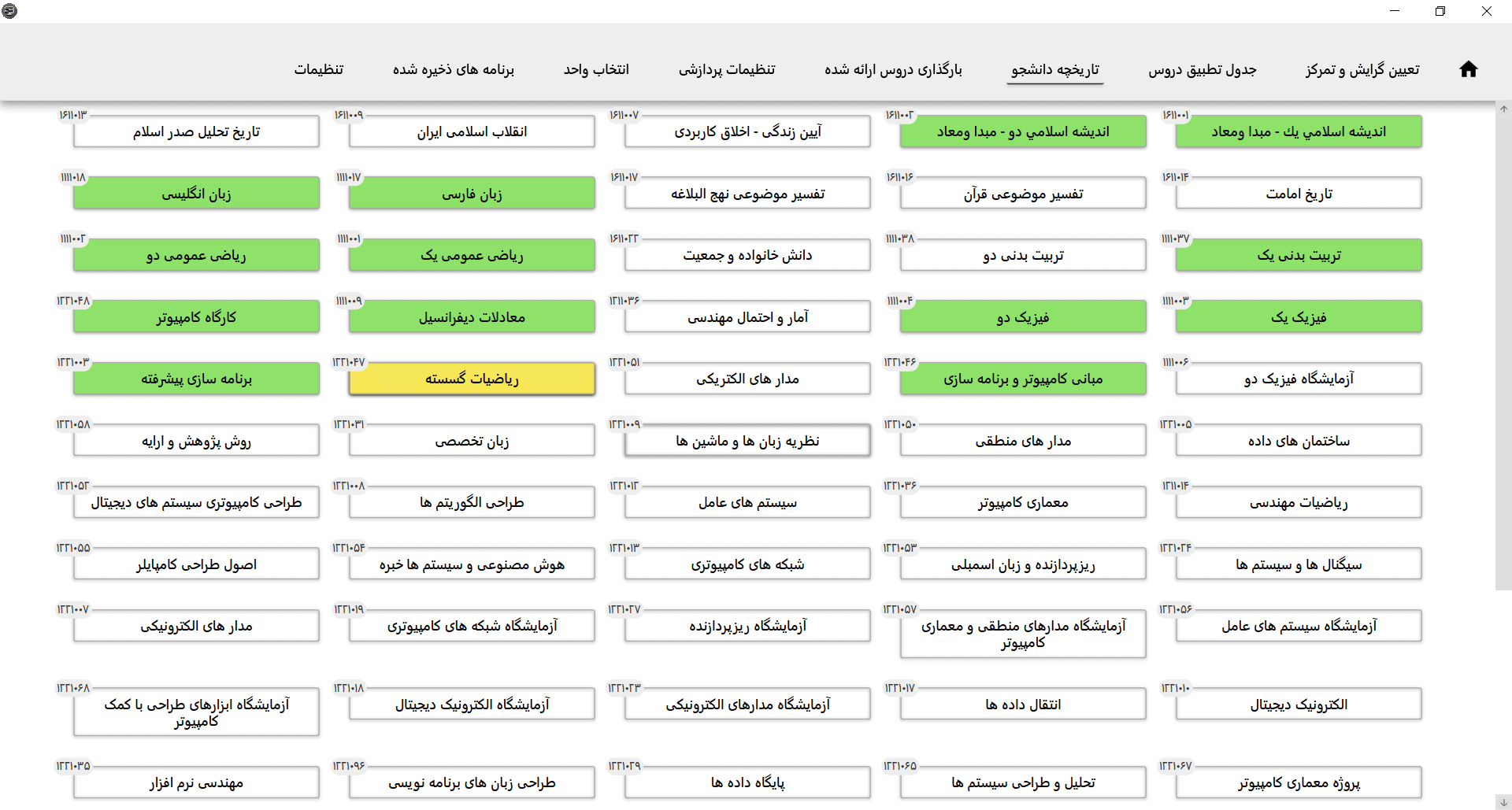
شکل 4-2 تصویری از قسمت­های تعین گرایش و تمرکز تخصصی

در شکل 4-2 بخش تعیین گرایش و تمرکز نشان داده شده است. در تصویر سمت راست کاربر گرایش تخصصی نرم­افزار و در تصویر سمت چپ کاربر تمرکز تخصصی الگوریتم و محاسبات را انتخاب کرده است.



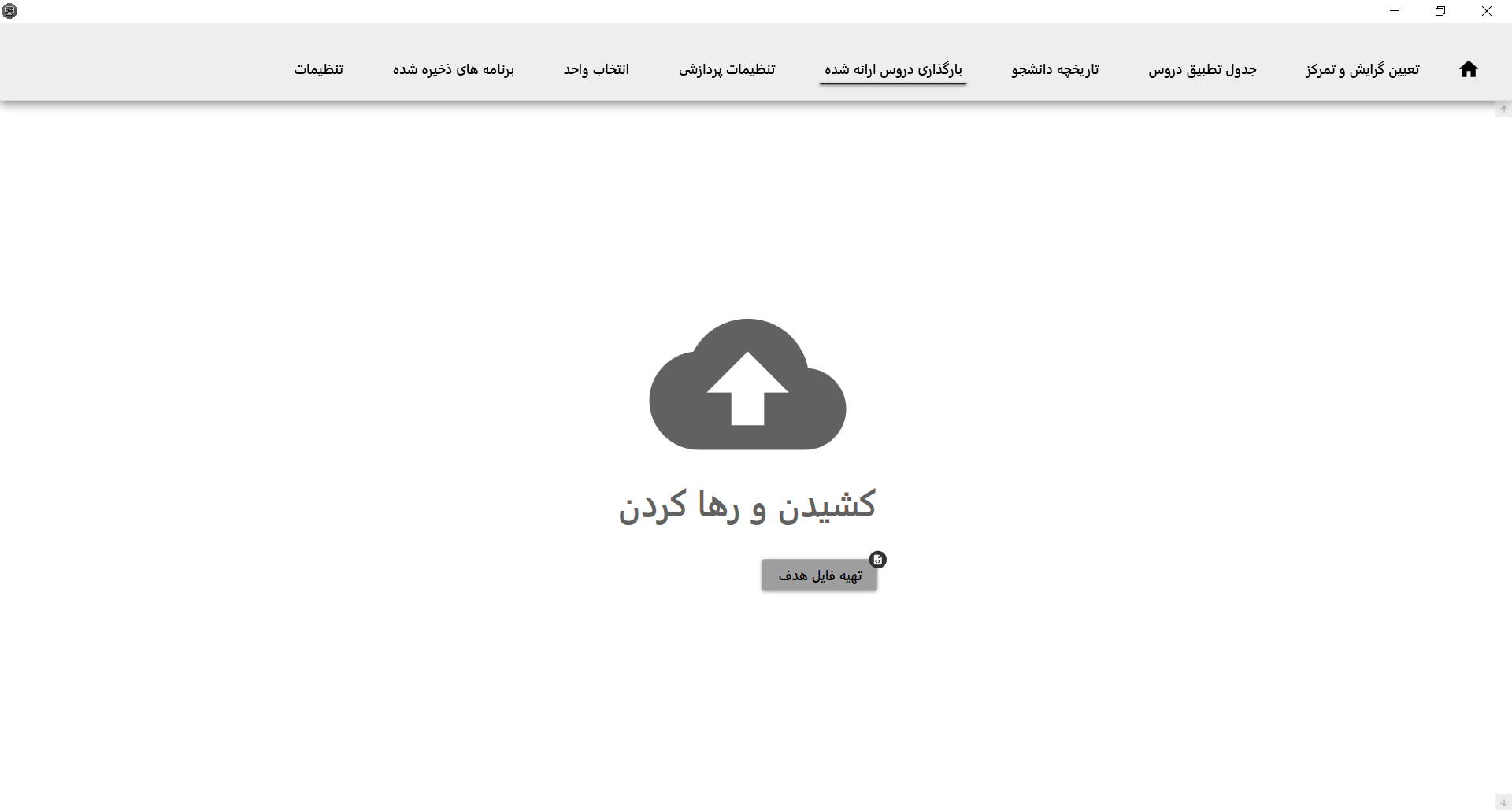
شکل 4-3 تصویری از بخش جدول تطبیق دروس

در شکل 4-3 بخش جدول تطبیق دروس نرم­افزار نشان داده شده است و داده­های آن بر اساس اطلاعات دانشگاه صنعتی نوشیروانی بابل پر شده است. خوب است به این نکته اشاره کنیم که جدول تطبیق دروس به تنهایی نمی­تواند ضامن سازگاری این نسخه در دانشگاه­های دیگر باشد چرا که همه دانشگاه­ها خود را ملزم به رعایت پیشنیاز­ها و همنیاز­های برنامه درسی مصوب وزارت علوم نمی­دانند بنابراین برای آنکه بتوانید این نسخه را در دانشگاهی که چنین شرایطی دارد استفاده کنید، باید داخل کد برنامه، قسمتی که ساختار توصیف کننده برنامه درسی مصوب وزارت علوم توصیف شده­است، پیشناز­ها و همنیاز­ها را بر اساس شرایط دانشگاه مورد نظر اصلاح کرد.



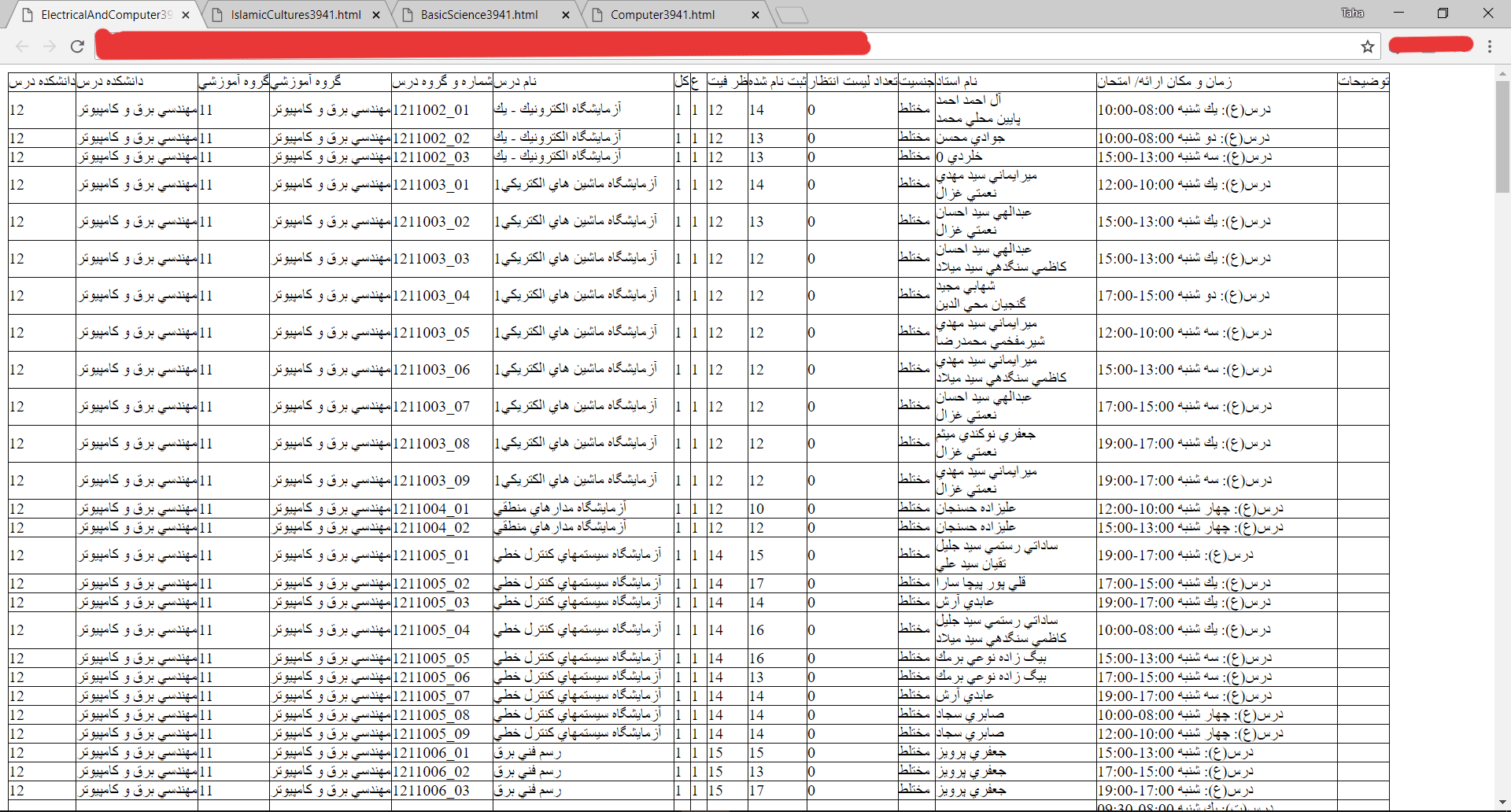
شکل 4-4 تصویری از بخش تاریخچه دانشجو

در بخش تاریخچه دانشجو درس­های سبز، درس­هایی هستند که دانشجو آنها را گذرانده است. درس­های زرد به این معنی هستند که دروسی را که این درس پیشنیاز آنها است می­توان با آن همنیاز کرد( مثلا دانشجویی که دوبار درسی را رد شده است طبق قوانین فعلی مجاز به انجام چنین کاری می­باشد) و سایر درس­ها سفید هستند.



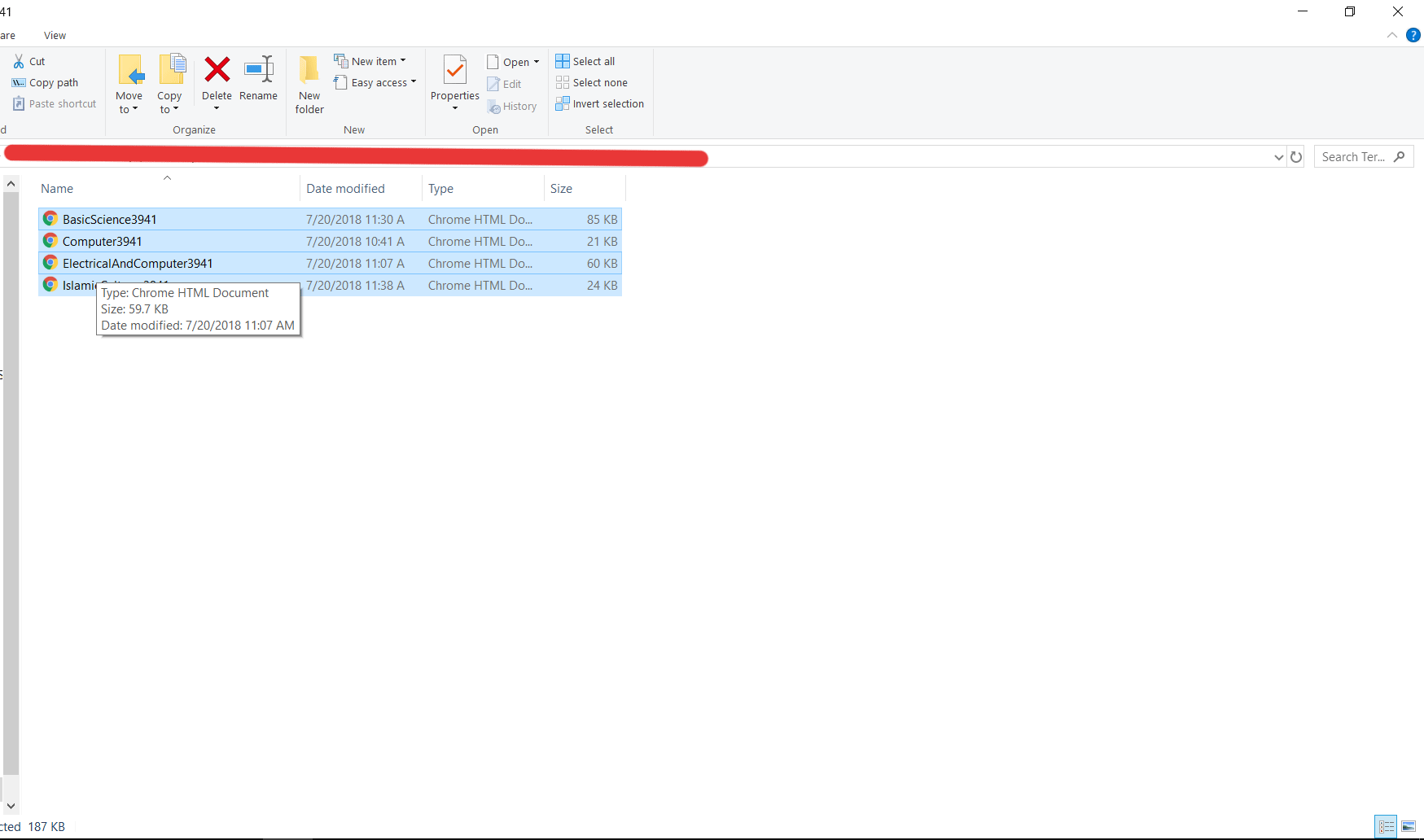
شکل 4-5 تصویری از بخش بارگذاری دروس ارئه شده

در بخش بارگذاری دروس ارائه شده، می­توانید فایل­های مربوط به برنامه دروس ارائه شده در ترم توسط دانشگاه را که در سیستم گلستان قابل دریافت است برای پردازش بارگذاری کنید.



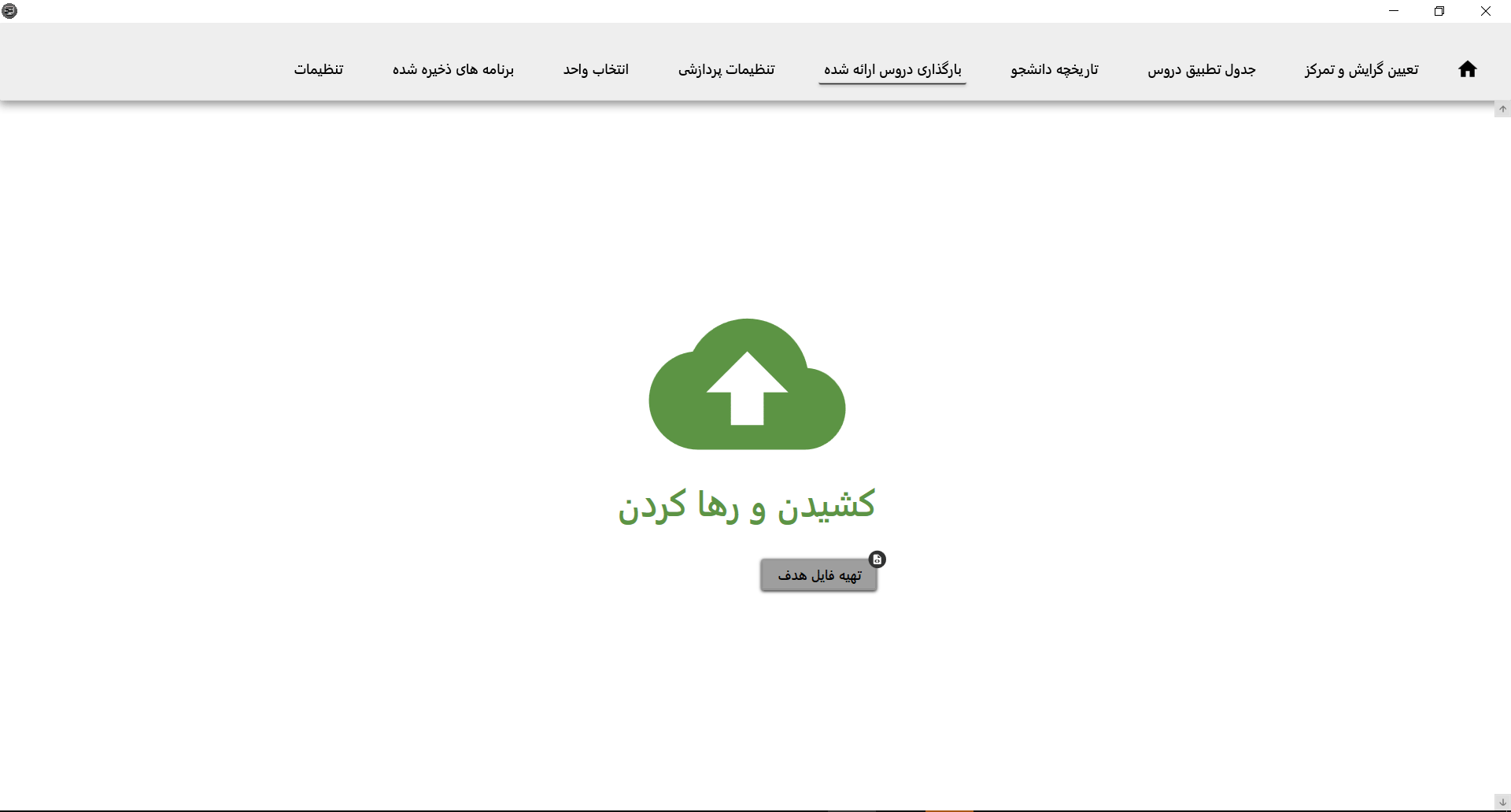
شکل 4-6 تصویری از نمونه فایل­های قابل دستیابی در سیستم گلستان

ابتدا باید برنامه­های مورد نیاز خود را در سیستم خود بصورت فایل­های اچ تی ام ال­ای ذخیره کنید. برای مثال ما در اینجا چهار فایل مربوط به دروس ارائه شده در ترم توسط برق و کامیپوتر، کامپیوتر، علوم پایه و معارف اسلامی را ذخیره کردیم.



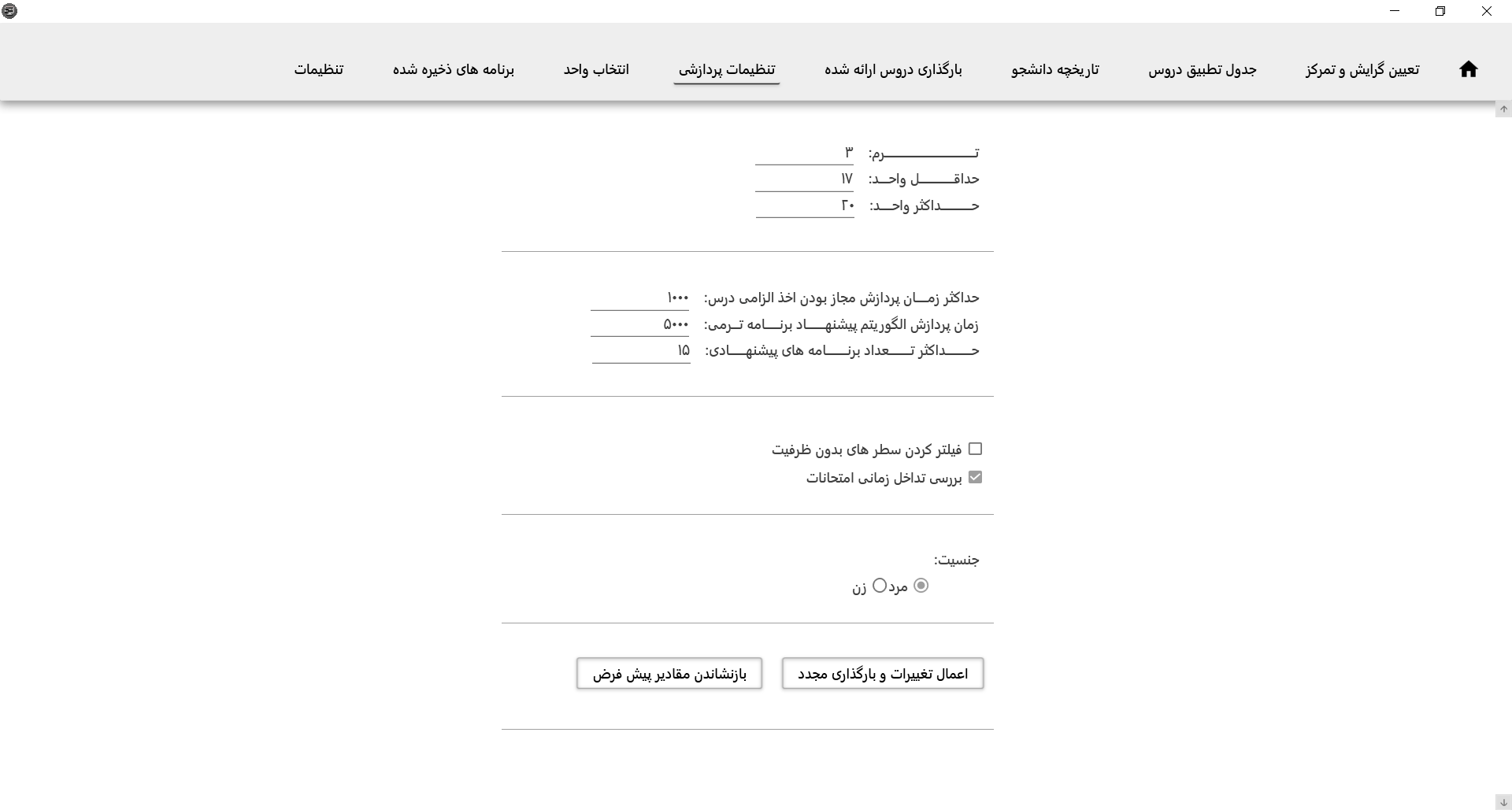
شکل 4-7 تصویری از فایل­های ذخیره شده مربوط به دروس ارائه شده در ترم توسط دانشگاه

سپس همه فایل­های مورد نیاز خود را انتخاب کرده با drag and drop آنها را به داخل شکل ابر مانند موجود در بخش بارگذاری دروس ارائه شده بکشید و رها کنید و در آخر بر روی گزینه تهیه فایل هدف کلیک کنید.



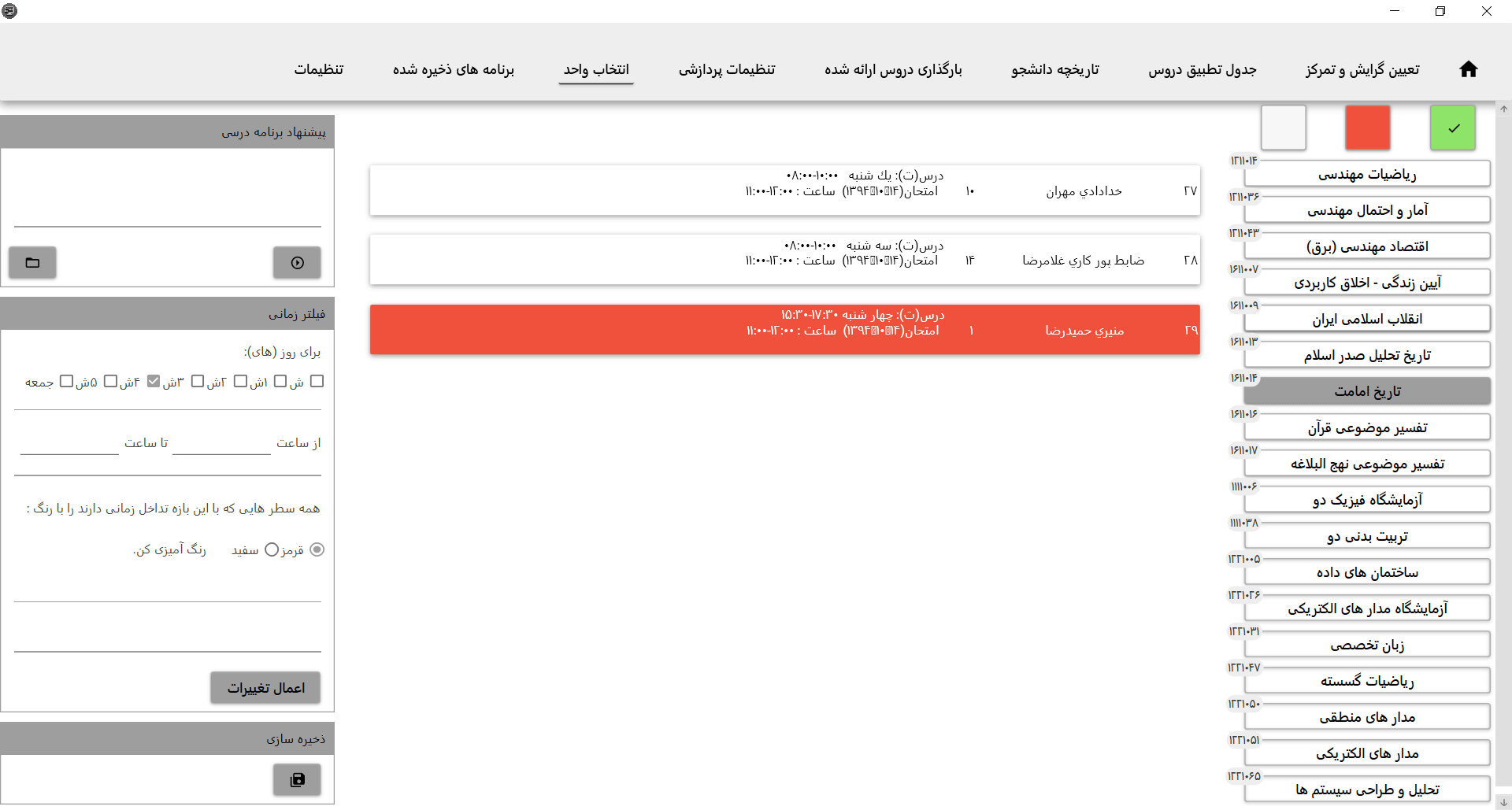
شکل 4-8 تصویری از بخش بارگذاری دروس ارائه شده پس از تهیه موفقیت آمیز فایل هدف

چنانچه فایل هدف با موفقیت تولید شود بخش کشیدن و رها کردن به رنگ سبز در می­آید.



شکل 4-9تصویری از بخش تنظیمات پردازشی

در بخش تنظیمات پردازشی، یک فرم تعبیه شده است که طی آن برخی از اطلاعات مورد نیاز برای اجرای الگوریتم را مشخص می­کند. در قسمت ترم می­توانید مشخص کنید به عنوان دانشجوی ترم چندم می­خواهید انتخاب واحد کنید. در قسمت حداقل و حداکثر واحد می­توانید مشخص کنید برنامه­هایی که می­خواهید الگوریتم به شما پیشنهاد دهد، حداقل و حداکثر چند واحد باشد (اگر دقیقا تعداد خاصی واحد را مد نظر دارید کافیست حداقل و حداکثر واحد را روی همان واحد مد نظرتان تنظیم کنید). فیلد بعدی مشخص می­کند برای اعتبارسنجی درس­های سبز حداکثر چقدر زمان بر حسب میلی ثانیه در نظر گرفته شود. فیلد زمان پردازش الگوریتم پیشنهاد برنامه ترمی مشخص می­کند پردازش الگوریتم بر حسب میلی ثانیه چقدر طول بکشد. فیلد حداکثر تعداد برنامه­های پیشنهادی مشخص می­کند الگوریتم حداکثر چند برنامه ترمی به شما پیشنهاد دهد. گزینه فیلتر کردن سطر­های بدون ظرفیت مشخص می­کند آیا سطر­هایی که ظرفیت آنها پر شده است فیلتر شوند یا خیر. گزینه بررسی تداخل زمانی امتحانات مشخص می­کند آیا علاوه بر تداخل نداشتن کلاس­ها تداخل نداشتن زمان امتحانات نیز بررسی شود یا خیر. در آخر با توجه به جنسیت می­توانیم سطر­هایی که از نظر جنسیت برای شما مجاز نمی­باشند را فیلتر کنیم.

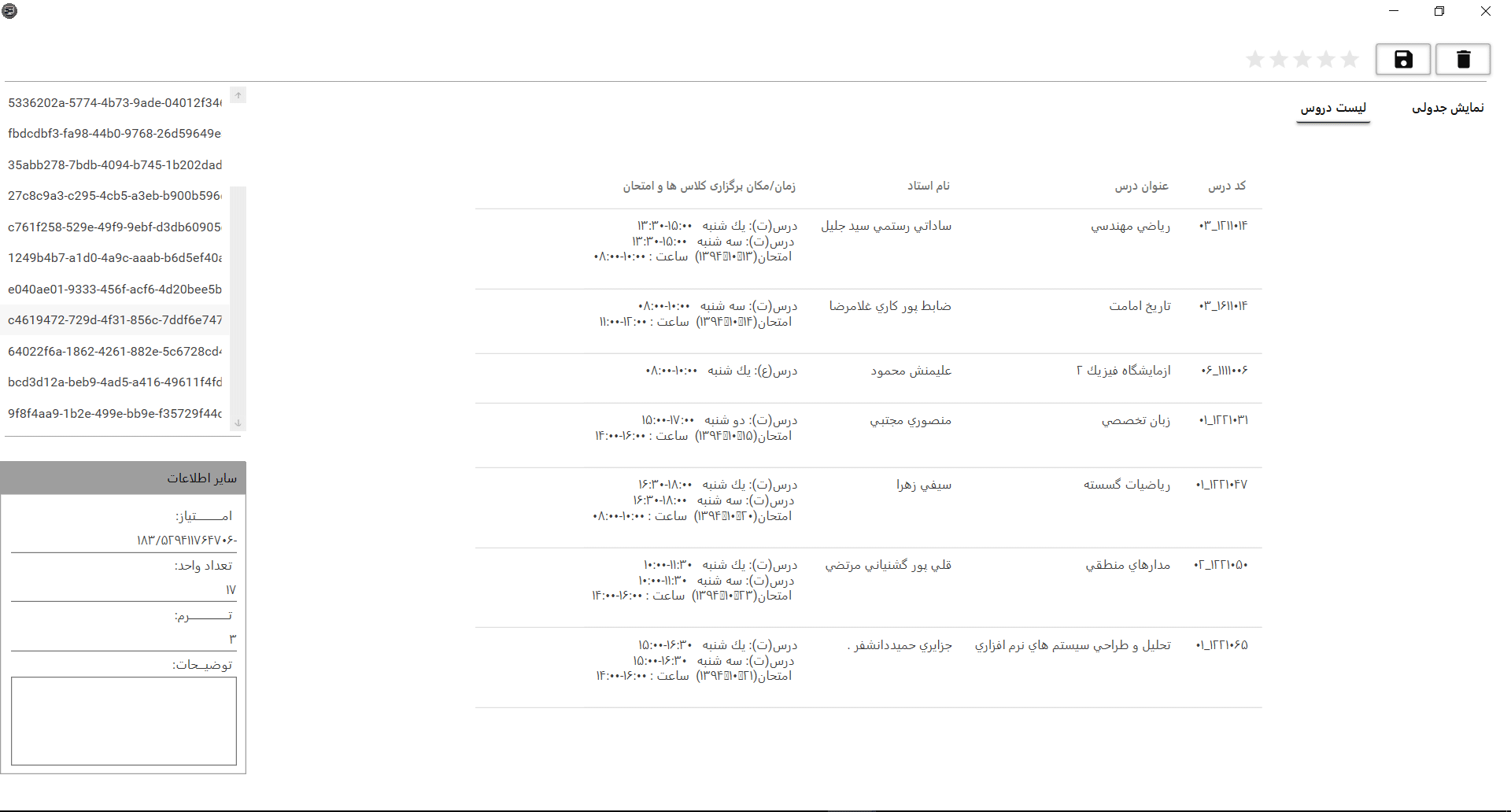


شکل 4-10 تصویری از بخش انتخاب واحد

در بخش انتخاب واحد درس­هایی که امکان اخذ آنها را دارید به همراه سطر­هایی که ممکن است آنها را اخذ نمایید نمایش داده می­شود. نام این دروس به همراه کد آنها در سمت راست این بخش نمایش داده می­شوند. بر روی هر درس که کلیک کنید، رنگ دکمه آن درس خاکستری می­شود و سطر­های آن به نمایش در می­آید.

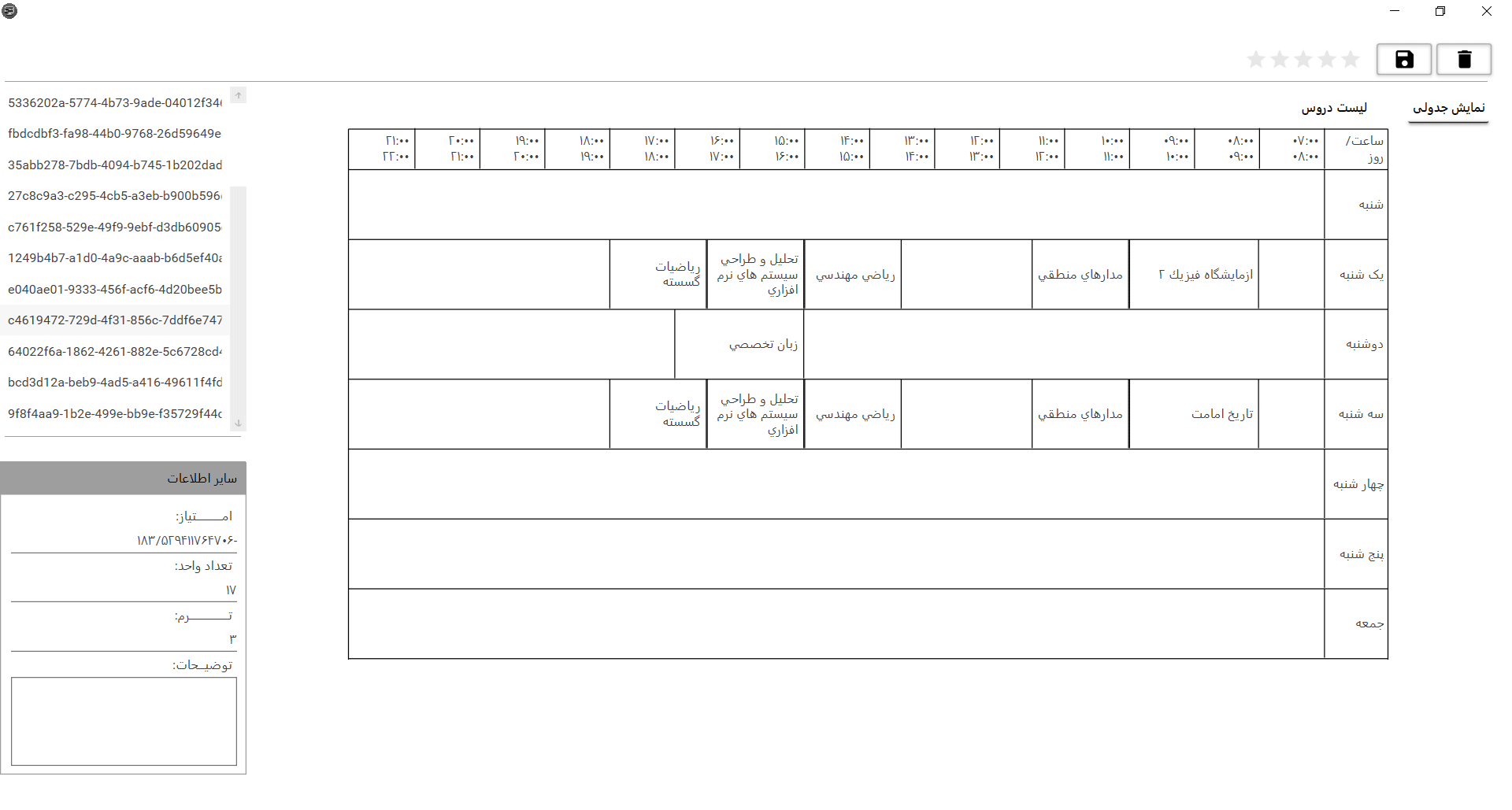
در بالای صفحه سمت راست، سه مربع قرار گرفته اند. با کلیک بر روی مربع اول از سمت راست می­توانید مشخص کنید درس سبز باشد(در برنامه های پیشنهادی حتما این درس وجود داشته باشد) یا خیر(بسته به شرایط درس بتواند اخذ شود یا نشود). با کلیک بر روی مربع قرمز رنگ می­توانید همه سطر­های سفید رنگ یک درس غیر سبز را به رنگ قرمز درآورید. و با مربع کناری می­توانید همه سطر­های قرمز یک درس را سفید کنید. علاوه بر این با دوباره کلیک بر روی هر سطر می­توانید در صورت مجاز بودن، رنگ سطر را از سفید به قرمز یا از قرمز به سفید تغییر دهید. سطر­های قرمز به این معنی هستند که در برنامه­های پیشنهادی الگوریتم نباید این سطر­ها حضور داشته باشند. در بخش مربوط به فیلتر زمانی می­توانید رنگ کردن سطر­ها را با شیوه­ای متفاوت انجام دهید. برای مثال فرض کنید بدلایلی مثلا کار نیمه وقت نمی­خواهید در زمان­های خاصی کلاس داشته باشید در اینصورت می­توانید بازه­های زمانی و روز­هایی که نمی­خواهید کلاس داشته باشید را در این بخش مشخص کرده و رنگ آنها را به کمک این ابزار قرمز کنید. اگر از بخش انتخاب واحد به بخش دیگری منتقل شوید یا نرم­افزار را بسته مجدادا اجرا کنید اطلاعات این بخش از دست خواهد رفت چنانچه می­خواهید در مراجعات بعدی به این بخش اطلاعات شما ذخیره بماند(تغییراتی که در رنگ سطر­ها و درس­ها داده­اید) می­توانید بر روی گزینه موجود در قسمت ذخیره­سازی کلیک کنید تا با این کار اطلاعات شما ذخیره شود. پس از خاتمه اجرا می­توانید برنامه­های پیشنهادی را با کلیک بر روی دکمه سمت چپ همین قسمت مشاهده کنید.

وقتی تنظیمات مورد نظر خود را انجام دادید، می­توانید از قسمت پیشنهاد برنامه درسی، بر روی دکمه سمت راست آن کلیک کنید تا الگوریتم پیشنهاد برنامه­های ترمی اجرا شود.



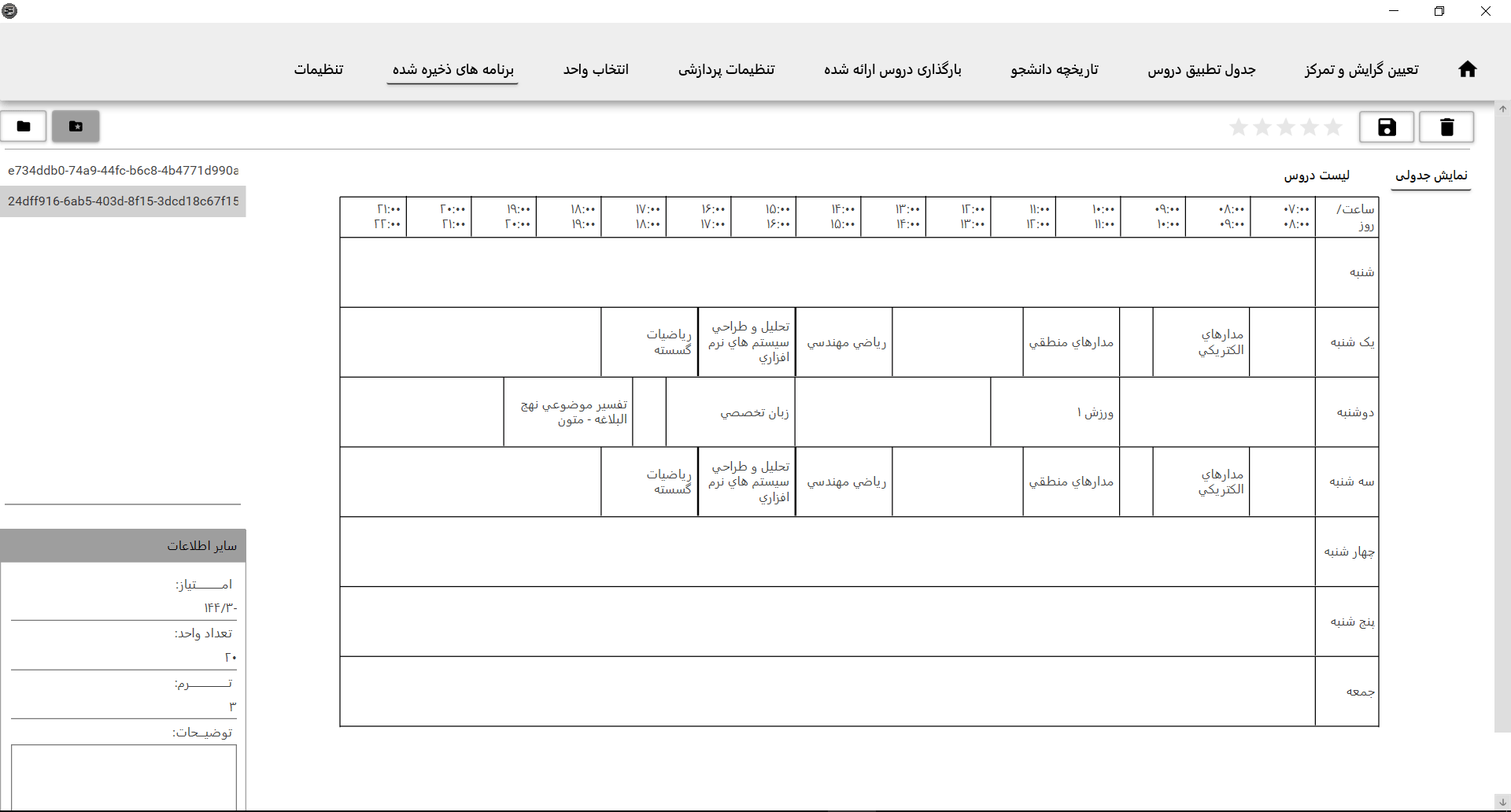
شکل 4-11تصویری از یک نمونه برنامه ترمی پیشنهاد شده توسط الگوریتم در نمایش لیست دروس

الگوریتم­های پیشنهاد شده توسط الگورتیم با دو شیوه **نمایش جدولی** و **لیست دروس** قابل نمایش هستند. در شیوه لیست دروس، درس­ها با ترتیبی قابل اخذ برای سیستم گلستان نمایش داده می­شوند.



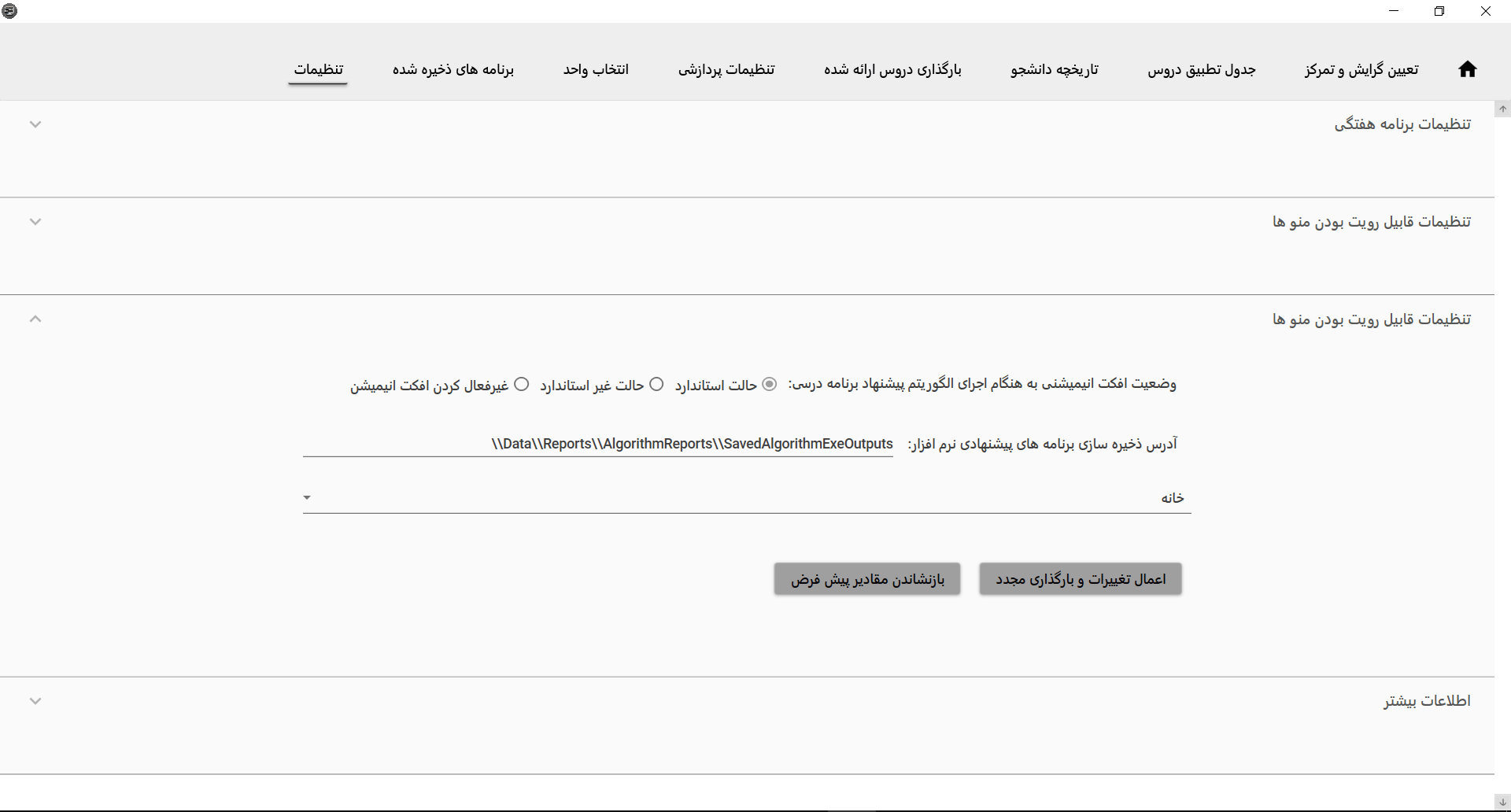
شکل 4-12 تصویری از یک نمونه برنامه ترمی پیشنهاد شده توسط الگوریتم در نمایش جدولی

در شیوه نمایش جدولی هم، برنامه پیشنهادی در قالب جدول برنامه هفتگی به نمایش در می­آید.



شکل 4-13تصویری از بخش برنامه­های ذخیره شده

در بخش برنامه­های ذخیره شده، می­توانید هم به برنامه­های پیشنهادی در آخرین اجرای الگوریتم و هم به برنامه­هایی که برای خود ذخیره کرده­اید دسترسی پیدا کنید.



شکل 4-14 تصویری از بخش تنطیمات

علاوه بر آنچه تا به اینجا گفته شد، این نسخه بخشی به نام تنظیمات دارد که از طریق آن می­توانید تنظیمات مربوط به نرم­افزار را انجام دهید.

4-3 نتایج حاصل از آزمایشات

در شکل­های 4-12 و 4-13 دو نمونه از برنامه­های ترمی پیشنهادی توسط الگوریتم مشاهده می­شوند. در اینجا نمونه‌های بیشتری را ارائه نمی­کنیم اما در مورد برخی از نکات در رابطه با نمونه­هایی که با آزمایش برای دوره تحصیلی یکی از دانشجویان در هر ترم بدست آمده، بحث می­کنیم.

برنامه­های پیشنهاد شده معمولا در عمل به گونه­ای بودند که، روز یا روز­هایی در برنامه هفتگی آزاد می­ماند(کلاسی نداشته باشند).

اغلب در عمل وقتی یک محدوده حداقل و حداکثر واحدی برای الگوریتم انتخاب می­کنیم بیشتر جواب­هایی که پیدا می­شوند متمایل به یکی از واحد­ها هستند مثلا اگر حداقل و حداکثر واحد را به ترتیب 17 و 19 در نظر گرفته و تعداد برنامه پیشنهادی را 15 در نظر بگیریم در 15 برنامه پیشنهادی تعداد زیادی از برنامه­ها مربوط به یکی از واحد­ها مثلا 18 هستند و برنامه­های پیشنهادی از سایر واحد­ها در اقلیت قرار می­گیرند. در فصل قبل وقتی معیاری را برای ارزیابی ارزش هر برنامه ترمی معرفی می­کردیم خاطر نشان کردیم که می­خواهیم برنامه­های پیشنهادی صرفا برای یک واحد خاص نباشند که این امر محقق شد و همینکه تعدادی اقلیت از واحد­های دیگر هم در برنامه­ها می­آیند نشانگر همین موضوع است. با این حال این اقلیت بودن خود شک برانگیز است چرا که این سوال را پیش می­آورد که در اکثریت چشمگیر قرار گرفتن یکی از واحد­ها در برنامه­های پیشنهادی به چه دلیل است؟ بنظر می­رسد پاسخ را باید در سه احتمال جستجو کرد. اول آنکه ممکن است معیار ارزیابی ارائه شده موجب این امر شده باشد. دوم اینکه ممکن است خود الگوریتم و شیوه گزینش آن موجب چنین امری شود. سوم این احتمال وجود دارد که در عمل محدودیت­هایی که توسط برنامه­ دروس ارائه شده توسط دانشگاه در ترم و شرایط دانشجو اعمال می­شوند واقعا ما را در وضعیتی قرار می­دهند که تعداد بخصوصی از واحد برای اخذ در آن ترم مناسب تر باشد. ممکن هم هست که ترکیبی از این موارد باعث چنین امری شده باشند. به هر حال تا به اینجا من نتوانسته­ام نتیجه گیری کنم که دلیل این اتفاق چیست و آیا اصلا این خوب است یا نه.

علاوه بر آنچه تا به اینجا گفته شد خوب است یادآور شویم که الگوریتم پیشنهاد برنامه­های ترمی براساس معیاری که طراح آن را مناسب می­دانسته توسعه یافته ولی شکی نیست که دیدگاه­ها و سلایق مختلف می­توانند معیار­های دیگری را برای یک برنامه، خوب بدانند. در آخر ارزیابی من از برنامه­های پیشنهادی این است که فکر می­کنم براساس معیار­هایی که من برای یک برنامه ترمی خوب می­دانم، الگوریتم برنامه­های بسیار خوب و همچنین متنوعی را در نتیجه آزمایشات انجام شده ارائه داده است.

4-4 خلاصه

در این فصل ابتدا با یک نسخه آزمایشی توسعه داده شده آشنا شدیم و بخش­های مختلف آنرا به همراه چگونگی بکار بردنشان توضیح دادیم. سپس به تشریح نتایجی که از برنامه­های پیشنهاد شده توسط الگوریتم بدست آمده بودند پرداختیم.

فصل پنجم:

نتیجه گیری­ها و پیشنهاد­ها

5-1 نتیجه گیری­ها

در این پروژه پس از مطالعه­ی برنامه­های درسی مصوب وزارت علوم به تحلیل و طراحی جهت ارائه راهکاری عملی و کارآمد برای پیشنهاد برنامه­های ترمی جهت انتخاب واحد دانشجویان مبادرت کردیم. سپس نسخه­ای آزمایشی در همین جهت پیاده­سازی شد که در عمل توانست برنامه­های ترمی مناسبی را در زمانی کوتاه و قابل تنظیم ارائه دهد. سعی شد تا قسمت تحلیل و طراحی اولیه، بسیار دقیق و جامع باشد که به نظرم این امر محقق شد با این وجود اما ساختاری که می­خواستم بعنوان توصیف کننده برنامه­های درسی ارائه دهم در حالت کلی شکست خورد و ساختار به ساختار توصیف کننده برنامه­های درسی وزارت علوم محدود شد. اما در نهایت فکر می­کنم پروژه خوبی بود هرچند نیاز به کار بیشتری دارد.

5-2 پیشنهاد­ها

برخی از پیشنهاد­­هایی که برای ادامه این پروژه می­توان ارائه داد را در ادامه لیست کرده­ایم:

* ارائه ساختار توصیف کننده برنامه­های درسی برای حالت عمومی و یا اصلاح همین ساختار فعلی
* افزودن سایر الویت­های باید و شاید
* گنجاندن امکان اصلاح پیشنیاز­ها و همنیاز­های دروس در برنامه کاربردی
* توسعه یک برنامه کاربردی که تولید ساختار توصیف کننده برنامه­های درسی را بصورت ویژوال تسهیل نماید.
* اضافه کردن امکاناتی جهت افزودن اسکریپت­ها یا تکه برنامه­ها در برنامه کاربردی با این ایده که هر شخص بتواند معیار­های خود در مورد برنامه ترمی خوب، توصیف کند.
* اضافه کردن امکان انتخاب واحد خودکار که در آن دانشجو برنامه­ای از برنامه­های پیشنهادی را انتخاب کند و برنامه کاربردی بصورت خودکار پس از تائید توسط دانشجو آن برنامه را در سیستم برایش اخذ نماید.