گزارش پروژه پایانی
ساختمان داده و الگوریتم

نام و نام خانوادگی: مهسا شیخی
شماره دانشجویی: 4023210164
تاریخ تحویل: 1404/04/20

ساختار نرم افزار

ساختار فایل‌های نرم افزار

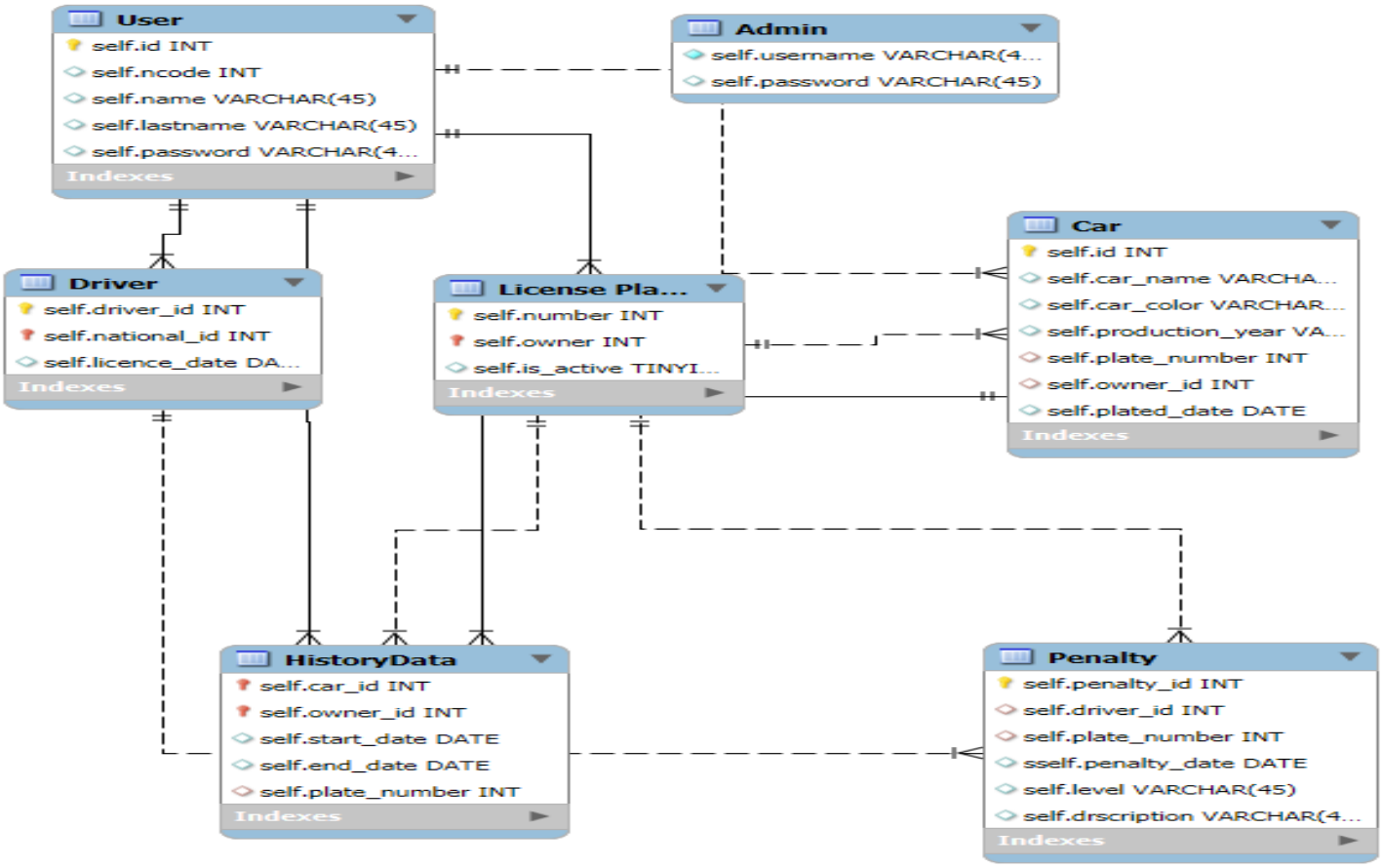
- **Project File:**
  - Apps:
    - Admin.py
    - Car.py
    - Driver.py
    - History.py
    - Penalty.py
    - Plates.py
    - User.py
  - DataSets:
    - __init__,py
    - Array.py
    - BSTHash.py
    - DoublyLinkedList.py
    - HashTable.py
    - Trie.py
  - Refrences
    - Pdf توضیحات پروژه
  - Tests
    - Cars,txt
    - Citycode.txt
    - Drivers.txt
    - Ownership_history.txt
    - Penalties.txt
    - Phase4.txt
    - Test_plates.txt

    - Users.txt
  - Module.py
  - Main.py

کلاس ها

- **کلاس شماره ۱ : [User]**
  - **نقش کلاس : نماینده کاربران سیستم. دارایی امکانات ورود و ثبت نام و تمام توابع خواسته شده در بخش پنل کاربر**
  - **دارای وابستگی به تمامی ساختمان داده ها و کلاس های مربوط به هر موجودیت مثل car , plates , history و ... جهت خواندن داده در هر ساختمان که دسترسی به آنها در بخش init کلاس به صورت self.className_database داده شده است.**
  - **بدون ارث بری از کلاس دیگر.**
- **کلاس شماره ۲ : [Admin]**
  - **نفش کلاس : نمایتده پنل مدیر در پروژه. دارای امکاناتی که در پروژه تحت عنوان پنل مدیریت تعریف شده اعم از پلاک کردن خودرو ، نمایش تمام خودرو ها ، کاربران ، پلاک های یک شهر ، راننده های یک شهر و... و همچنین شامل ابزار هایی که در توابع اصلی استفاده شده.**
  - **دارای وابستگی به تمامی ساختمان داده ها و کلاس های مربوط به هر موجودیت مثل car , plates , history و ... جهت خواندن داده در هر ساختمان که دسترسی به آنها در بخش init کلاس به صورت self.className_database داده شده است.**
  - **بدون ارث بری از کلاس دیگر.**
- **کلاس شماره ۳ : [Car]**
  - **شامل شی تعریف شده برای نگهداری اطلاعات هر خودرو در سیستم مانند owner id , plate number ، رنگ خودرو ، نام و سال تولید خودرو می باشد.**
- **کلاس شماره ۴ : [LicensePlate]**
  - **شامل شی تعریف شده برای نگهداری اطلاعات هر پلاک مانند شماره پلاک ، شناسه مالک پلاک و یک متغیر is_active که دارای مقادیر True , False است و جهت مشخص کردن فعال یا غیر فعال بودن هر پلاک قرار داده شده.**

- **کلاس شماره ۵ : [Penalty]**
  - **شامل شی تعریف شده برای نگهداری اطلاعات داده تخلف کاربران مانند شماره پلاک ، شناسه راننده ، شدت تخلف و توضیحات علت هر تخلف.**
- **کلاس شماره ۵ : [Driver]**
  - **شامل شی تعریف شده برای نگهداری اطلاعات داده هر راننده مانند شناسه کاربری (National id) ، شناسه راننده و شماره پلاک ثبت شده که ساخت آن توسط پنل مدیریت هنگام اعطای مجوز رانندگی به یک کاربر انجام می شود.**
- **کلاس شماره ۶ : [HistoryData]**
  - **شامل شی تعریف شده برای نگهداری اطلاعات مالکیت هر خودرو. مالکیت خودرو در صورت خرید و فروش تغییر می کند و این شی شامل اطلاعاتی مثل تاریخ شروع و پایان مالکیت ، شماره پلاک خودرو در این بازه ، شناسه مالک (National id) و شناسه خودرو است.**
- **کلاس شماره ۷ : [generator]**
  - **شامل یک تابع __str__ جهت generate کردن خروجی هر تابعی که به عنوان ورودی دریافت می کند.**
  - **کاربرد آن نشان دادن پیغام های ورودی و خروجی به کاربر در ترمینال است.**



ساختمان های داده و الگویتم‌ها

ساختمان های داده

- ساختمان داده ۱ [HashTable]:
  - جهت ذخیره ماشین ها. به دلیل اینکه درج خودرو ها بر اساس شناسه ی یکتای تولید شده به صورت تصادفی است، توزیع خوبی روی جدول رقم می زند . جدول به صورت پویا با linear probing پیاده سازی شده. دسترسی ، درج ، حذف و جست و جوی موفق این داده ها با احتساب هزینه prob و بدون expand، 1/1-alpha می باشد. و در صورت expand هزینه زمانی O(n) دارد.
  - DSACourseFinalProject/DataSets/HashTable.py
- ساختمان داده ۲ [Trie]:
  - برای ذخیره سازی کاربران و راننده ها بر اساس شناسه کاربری آنها که یک عدد با طول ثابت k است و هزینه درج و حذف و جست و جو را ثابت نگه می دارد.
  - DSACourseFinalProject/ DataSets/Trie.py
- ساختمان داده ۳ [BSTHash]:
  - استفاده از یک درخت bst برای جلوگیری از collisionها . ایده پشت این ساختمان داده خوشه بندی پلاک ها بر اساس شهر بود و همچنین کاهش هزینه جست و جو به O(1 +n).
  - DSACourseFinalProject/ DataSets/BSTHash.py
- ساختمان داده ۴ [DoublyLinkedList]:
  - برای ذخیره سازی تاریخچه خرید و فروش و جرایم. عمده عملیات روی این داده ها insert است و می تواند با استفاده از درج در ابتدای لیست با هزینه O(n) ساختمان داده ی پویا و خوبی برای داده هایی که تعدادشان افزایش می یابد و عمدتا درج و پیمایش دارد (هیستوری ها) باشد.
  - DSACourseFinalProject/DataSets/DpublyLinkedList.py
- ساختمان داده ۳ [Array]:
  - سادگی پیاده سازی برای داده هایی که درج و حذف ندارند و مقدار مشخص و بدون تغییری دارند همچنین دسترسی سریع به شهر ها با O(n)
  - DSACourseFinalProject/DataSets/Array.py

الگوریتم ها

- الگوریتم شماره ۱
توضیحات: [الگوریتم تولید پلاک با استفاده از تابع random برای تولید اعداد و حروف تصادفی، بررسی اعتبار پلاک با استفاده از تابع کپسوله شده _is_valid_plate_number و جلوگیری از تکراری بودن با حلقه for روی عدد تولید شده و بررسی تعداد تکرار هر عنصر در آن با تابع _has_repeted_digit در صورتی که تعداد تکرار اعداد ۵ تا بود یعنی اعداد مشابه هستند و یا صعودی نزولی بودن اعداد با استفاده از حلقه for روی اعداد در صورتی که تمام اعداد از عدد بعد از خودشان کوچکتر بودند ارور صعودی و در صورتی که تمام اعداد از عدد بعد از خود بزرگتر بودند ارور نزولی بودن می دهد هزینه زمانی به اندازه طول پلاک تولید شده]

  - DSACourseFinalProj/Apps./User.py/User.license_plate_generator

- الگوریتم شماره ۲
توضیحات: [الگوریتم هش کردن رمز عبور با استفاده از یک salt ثابت و تبدیل تمام کرکتر های رمز و سالت به عدد با تابع ord و جمع زدن آنها]

  - DSACourseFinalProject/Apps/User.py/user._password_hash_function

  - الگوریتم شماره **3**
توضیحات: [الگوریتم پیمایش تمام داده های trie به صورت بازگشتی با استفاده از جست و جوی اول عمق یا dfs به این صورت که از ریشه شروع می کند و تابع بازگشتی_recersive_traverse را برای هر node صدا میزند. مسیر هر گره را در prefix ذخیره می کند اگر داده بعدی گره داشته باشد مسیر ان را اضافه می کند و اگر داده نداشته باشد ریترن می کند]

  - DSACourseFinalProject/DataSets/Trie.py/Trie.traverse

- الگوریتم شماره ۴

توضیحات: [ساختمان داده ترکیبی ساخته شده ی BSTHash به این صورت است که تمامی اعمال مربوط به bst را در خود دارد. در توابع مربوط به جدول hash و با پیدا کردن خانه مربوطه از جدول با استفاده از تابع hash اعمال bst را روی آن خانه جدول صدا میزند. به عنوان مثال برای درج عنصر ابتدا تابع hash خانه مناسب با آن را پیدا می کند سپس تابع درج bst به ریشه ی آن خانه ی جدول صدا زده می شود.]

پیاده‌سازی نرم‌افزار

ساختمان داده‌های اصلی

- ساختمان داده ۱ [HashTable]:
  - شامل یک open hash و شیوه probing به صورت linear جهت ذخیره سازی خودرو ها و درج آنها بر اساس car _id.
  - ذخیره شده در آدرس DSACourseFinalProject/DataSets/HashTable.py
- ساختمان داده ۲ [Trie]:
  - شامل یک Trie Tree و Trie Node برای ذخیره سازی کاربران و راننده ها و درج آنها بر اساس Naitonal id.
  - ذخیره شده در آدرس DSACourseFinalProject/ DataSets/Trie.py
- ساختمان داده ۳ [BSTHash]:
  - شامل یک جدول hash که در هر خانه از table آن یک درخت bst قرار می گیرد جهت ذخیره پلاک ها. پلاک ها بر اساس شهر وارد خانه مربوط در جدول میشوند و بر اساس شماره پلاک در درخت مربوط به آن خانه ی جدول درج می شوند.
  - ذخیره شده در آدرس DSACourseFinalProject/ DataSets/BSTHash.py
- ساختمان داده ۴ [DoublyLinkedList]:
  - شامل یک لیست پیوندی دو طرفه و node برای ذخیره جریمه ها و تاریخچه خرید و فروش.
  - ذخیره شده در آدرس DSACourseFinalProject/DataSets/DpublyLinkedList.py
- ساختمان داده ۳ [Array]:
  - شامل یک جدول آرایه برای ذخیره سازی کد شهر ها.
  - ذخیره شده در آدرس DSACourseFinalProject/DataSets/Array.py

عملکردها

- فاز اول پنل کاربر در User.py:
  - ثبت نام و ورود کاربر:
    - از طریق دریافت شناسه و رمز عبور از کاربر با جست و جو در اطلاعات کاربران در صورتی که نام کاربری و رمز کاربر درست باشد به کاربر پیغام مرتبط می دهد. در صورتی که کاربر در سیستم ثبت نباشد با تابع user_register در پنل کاربر در کلاس User.py با گرفتن تمام اطلاعات لازم ثبت را انجام و اطلاعات کاربر را به ساختمان داده مرتبط با هزینه O(k) که k تعداد رقم های کد ملی است اضافه می کند. و در نهایت با توجه به هزینه جست جوی کاربر با همان O(k) هزینه ی زمانی ثابت دارد.
  - ایجاد پلاک:
    - با گرفتن نام شهر از طریق license_plate_generator و جست جوی شهر در ساختمان داده مربوط به آن و همچنین گرفتن شناسه کاربر یک شماره پلاک با رعایت قوانین مربوطه برای کاربر به نام او تولید می کند. هزینه زمانی آن با توجه به جست و جوی شهر ها با O(100) و درج در ساختمان داده پلاک ها O(1 + logn) تقریبا همان logn است.
  - مشاهده خودرو های ثبت شده: تابع درست کار نمی کند.
    - تابع show_user_cars با گرفتن شناسه کاربر با پیمایش تمام داده های موجود در ساختمان داده ماشین ها تمامی خودرو های متعلق به او را نشان می دهد که هزینه زمانی O(n)دارد.
  - مشاهده پلاک های کاربر:
    - تابع show_users_platelicense با گرفتن شناسه کاربر با پیمایش تمام داده های موجود در ساختمان داده پلاک ها تمامی خودرو های متعلق به او را نشان می دهد که هزینه زمانی O(n)دارد.
- فاز اول پنل مدیریت در Admin.py:
  - پلاک کردن خودرو: تابع به درستی کار نمیکند.
    - تابع plated_a_car با گرفتن شماره پلاک و اطلاعات خودرو در صورت وجود شماره پلاک با جست و جویی با هزینه O(Logn) خودرو جدید را در ساختمان داده با هزینه O(prob + 1)درج می کند.
  - مشاهده تمامی خودرو ها:
    - تابع show_all_cars با پیمایش تمام داده های موجود در ساختمان داده ماشین ها تمامی خودرو هارا نشان می دهد که هزینه زمانی O(n)دارد.

- مشاهده تمامی کاربران:

  - تابع show_all_users با پیمایش تمام داده های موجود در ساختمان داده کاربران تمامی کاربران ثبت شده را نشان می دهد که هزینه زمانی O(k)دارد و k طول شناسه کاربری است.

- فاز دوم پنل مدیریت در Admin.py:

  - مشاهده تمامی پلاک های یک شهر:

    - تابع show_plates_of_a_city با گرفتن نام شهر و تبدیل آن به کد شهر با get_citycode_from_cityname_ با رفتن به خانه مربوط به شهر در table ساختمان داده و پیمایش درخت مربوط به آن با هزینه O(logn) پلاک های مربوط به شهر را نشان میدهد.

  - مشاهده خودرو های یک شهر:

    - تابع show_cars_of_a_city با پیمایش تمام داده های موجود در ساختمان داده ماشین ها تمامی خودرو هایی که شماره پلاک آنها متعلق به شهر مربوطه هستند را نشان می دهد که هزینه زمانی O(n)دارد.

  - جست و جوی خودرو های ثبت شده در بازه زمانی:

    - تابع search_cars با گرفتن دو بازه ورودی و پیمایش تمام داده های موجود در ساختمان داده ماشین ها تمامی خودرو هایی که تاریخ آنها متعلق به بازه وارد شده باشد نشان می دهد که هزینه زمانی O(n) دارد.

  - مشاهده مالکان خودروی یک شهر: تابع درست کار نمی کند.

    - تابع show_car_owners_of_a_city با گرفتن نام شهر و پیمایش خودرو ها، برای خودرو هایی با پلاک مربوط به آن شهر، شناسه کاربر را با هزینه O(k) جست و جو و اطلاعات آنهارا نمایش میدهد. هزینه نهایی O(n + K) می باشد.

- فاز سه پنل کاربر در User.py:

  - مشاهده نمره منفی راننده:

    - تابع show_users_negative_score با گرفتن شناسه کاربری و شناسه راننده، با پیمایش اطلاعات جریمه ها تمامی جریمه های مربوط به راننده را جمع

    - می زند که کل کار هزینه O(n) دارد.

- مشاهده تاریخچه جرایم:
  - تابع show_user_penalties_based_driverid با گرفتن شناسه راننده تمامی اطلاعات جرایم را با O(n) پیمایش می کند و جریمه های مربوط به راننده را نشان می دهد.
- مشاهده تاریخچه جرایم یک پلاک:
  - تابع show_user_penalties_based_platenumber با گرفتن یک شماره پلاک تمامی اطلاعات جرایم را با هزینه زمانی O(n) پیمایش می کند و جریمه های مربوط به آن پلاک را نشان می دهد.
- اطلاعات تاریخچه یک پلاک:
  - تابع history_of_licenseplate با گرفتن یک شماره و جست و جوی خودروی مربوط به آن پلاک با هزینه زمانی O(n)، تمامی اطلاعات مربوط به آن پلاک و خودرو را با O(n) پیدا می کند نشان می دهد. در نهایت هزینه زمانی O(2n) دارد.

- فاز سه پنل مدیریت در Admin.py:
  - تاریخچه خرید و فروش یک خودرو:
    - تابع show_ownership_history با گرفتن شناسه خودرو و پیمایش تمام اطلاعات خرید و فروش ، تمامی اطلاعات مرتبط با خودرو را با هزینه زمانی O(n) نشان می دهد.
  - مشاهده رانندگان ثبت شده:
    - تابع show_all_drivers با پیمایش تمام داده های موجود در ساختمان داده راننده ها تمامی راننده ها را نشان می دهد که هزینه زمانی O(n) دارد.

  - تغییر مالکیت یک پلاک: تابع فقط پیاده سازی شده و تست نشده.
    - تابع change_plate_owner با گرفتن شماره پلاک جدید برای یک خودرو در صورتی که پلاک وجود داشته باشد(جست و جوی موفق با هزینه O(logn)) و متعلق به خودروی دیگری نباشد(با هزینه O(n)) شماره پلاک خودرو را عوض می کند و اطلاعات جدید را به اطلاعات خرید و فروش اضافه می کند.
- ما بقی توابع فاز سه پنل مدیریت و فاز ۴ پیاده سازی نشده اند.

چالش ها

چالش هایی که در این پروژه با آن مواجه شدید؟ **انتخاب ساختمان داده مناسب و بهینه برای پلاک ها و هندل کردن تکه تکه پیاده کردن پروژه در فایل های مختلف و سپس استفاده از همه آنها در main.py**

چالش هایی که در پیاده سازی با آن مواجه شدید؟ **پیاده سازی BSTHash و آماده سازی منو ها و نمایش دادن آنها و صدا زدن توابع در منو.**

نظر شما در مورد پروژه چیست؟

منابع : تمامی منابع مرتبط با پایتون در سراسر اینترنت، stackoverflow ، youtube و github

لینک github پروژه:

https://github.com/TheMahsami/DSACourseFinalProject