دانشکده مهندسی

پایان‌نامه کارشناسی ارشد

گروه مهندسی کامپیوتر

تحلیل، طراحی و پیاده‌سازی وب سایت مربوط به زنجیره تامین و شبکه جهانی حمل و نقل- بخش بلاکچین

نگارنده:

سارا بلوری بزاز

استاد راهنما:

دکتر عباس رسول‌زادگان

زمستان ۱۴۰۱


بسم الله الرحمن الرحيم

اصالت نامه

فرم ارزشیابی

تقدیم به

مقدس‌ترین واژه‌ها در لغت‌نامه دلم

مادر مهربانم که زندگیم را مدیون مهر و عطوفت او هستم

پدرم، مهربانی مشفق، بردبار و حامی

تقدیر و تشکر

حمد و سپاس سزاوار خداوندی است که مرا نعمت هستی بخشید و در مسیر آموختن علم قرار داد. در این مسیر با اساتیدی فرهیخته، صبور و با اخلاق آشنایم ساخت. هر چند در مقام قدردانی از زحمات ایشان زبان قاصر و دست ناتوان است، اما بر خود لازم می‌دانم از زحمات و راهنمایی‌های استاد گرانقدر جناب آقای دکتر عباس رسول‌زادگان قدردانی و تشکر نمایم چرا که بدون راهنمایی‌ها و دلسوزی‌های ایشان گردآوری این پایان‌نامه امکان‌پذیر نبود. همچنین از راهنمایی‌ها و کمک‌های همه اعضای محترم آزمایشگاه کیفیت نرم‌افزار نیز صمیمانه کمال تشکر و قدردانی را دارم.

بسمه تعالی

مشخصات رساله/پایان نامه تحصیلی دانشجویان

دانشگاه فردوسی مشهد

عنوان رساله/پایان نامه: تحلیل، طراحی و پیاده‌سازی وب سایت مربوط به زنجیره تامین و شبکه جهانی حمل و نقل- بخش بلاکچین

نام نویسنده: سارا بلوری بزاز

نام استاد(ان) راهنما: جناب آقای دکتر عباس رسول‌زادگان

نام استاد(ان) مشاور: --

دانشکده : مهندسی	**گروه:کامپیوتر**	**رشته تحصیلی: نرم افزار**
تاریخ تصویب:		**تاریخ دفاع:**
مقطع تحصیلی: کارشناسی ارشد		**تعداد صفحات:**

چکیده رساله/پایان‌نامه:

کلید واژه: **بلاکچین، ذخیره‌سازی امن داده، سیستم توزیع شده، قرارداد هوشمند، چارچوب هایپرلدجر فابریک**	**امضای استاد راهنما: دکتر عباس رسول‌زادگان** **امضا**

چکیده

فهرست مطالب

فهرست شکل‌ها

فهرست جدول‌ها

۱- مقدمه

۲- پیشینه

۳- راهکار پیشنهادی

هدف روش پیشنهادی، ذخیره‌سازی امن داده‌ها در شبکه بلاکچین است. بدین منظور سعی شده است برای به دست آوردن روش مناسب جهت ذخیره‌سازی، دو روش مختلف پیاده‌سازی شود. در ابتدا، ذخیره‌سازی داده‌های یک سیستم حمل و نقل جهانی تحت وب در شبکه بلاکچین صورت گرفته است. در انتها، کار انجام شده از حالت خاص منظوره به حالت عام منظوره تبدیل شده است. پیاده‌سازی روش‌های

پیشنهادی از دو بخش کاربر و سرور تشکیل شده که در ادامه، به جزئیات پیاده‌سازی هر یک از روش‌ها پرداخته شده است.

۳-۱- پیاده‌سازی پروژه – سمت کاربر

در پروژه حمل و نقل، امکان عقد قرارداد در خدمات مختلف برای کاربران مهیا شده است. این خدمات شامل موارد حمل و نقل دریایی، ریلی، زمینی، هوایی و چند وجهی است که هر یک شامل زیرخدمات مختلفی می‌شوند که در

خدمت‌ها	زیرخدمت‌ها								
دریایی	Chartering	Bulk	Line	FCL	LCL	Broker	Sale Container	Shipyard	Ship Chandler
ریلی	Expeditor			Owner Wagons			Rail Industry		
زمینی	Full Truck Loading			Less Truck Loading			Drivers		
هوایی	Air Cargo						Courier		
چند وجهی	Freight Forwarding						Logistic		

جدول۳-۳-۱ زیر آورده شده است:

جدول۳-۳-۱: خدمات و زیرخدمات در پروژه حمل و نقل

خدمت‌ها	زیرخدمت‌ها								
دریایی	Chartering	Bulk	Line	FCL	LCL	Broker	Sale Container	Shipyard	Ship Chandler
ریلی	Expeditor			Owner Wagons			Rail Industry		
زمینی	Full Truck Loading			Less Truck Loading			Drivers		
هوایی	Air Cargo						Courier		
چند وجهی	Freight Forwarding						Logistic		

در زیرخدمات FCL و Chartering مربوط به خدمت دریایی و Air مربوط به خدمت هوایی، تکنولوژی بلاکچین جهت ذخیره‌سازی قراردادها، پیاده‌سازی شده است. در این زیرخدمت‌ها، کاربران بعد از نهایی کردن قراردادهای خود، می‌توانند قراردادهای مذکور را در نرم افزار تحت بلاکچین مشاهده کنند. سپس در صورت تمایل، هر کاربر می‌تواند قرارداد خود را با کمک کلید خصوصی خود امضا کند و آن را وارد بلاکچین کند. با اضافه شده قرارداد به بلاکچین، قرارداد ذخیره و غیرقابل ویرایش خواهد شد. نهایی شدن قرارداد بدین معناست

که طرفین قرارداد بر سر تعدادی از مفاد خاص قرارداد به توافق برسند و در سایت، قرارداد اولیه خود را نهایی کنند.

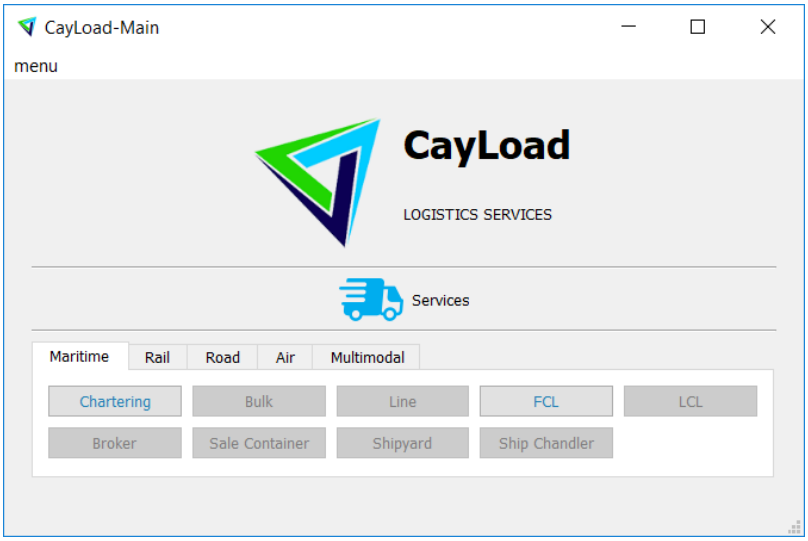
در امضا کردن قراردادها، لازم است که برای طرفین قرارداد کلید عمومی تولید شده باشد (به عبارت دیگر حداقل یکبار در نرم افزار بلاکچین وارد شده باشند) تا اجازه امضا کردن قرارداد به طرفین داده شود؛ در غیراینصورت امکان امضا کردن قرارداد در نرم افزار از طرفین گرفته می‌شود.

در این طرح، از آنجایی که پروژه یک پروژه خصوصی است، بلاکچین آن از نوع خصوصی می‌باشد و اعضای موجود در بلاکچین توسط سرور مرکزی سایت مورد احراز هویت قرار می‌گیرند. به عبارت دیگر، تنها کاربرانی که در وبسایت مربوطه ثبت نام کرده باشند امکان استفاده از نرم افزار بلاکچین را دارد؛ چرا که در ابتدای استفاده از نرم افزار لازم است کاربر احراز هویت کند.

از آنجایی که پیاده سازی زیرخدمت‌های FCL و Chartering با Air متفاوت است، جزئیات پیاده سازی هر یک به صورت جداگانه شرح داده شده است.

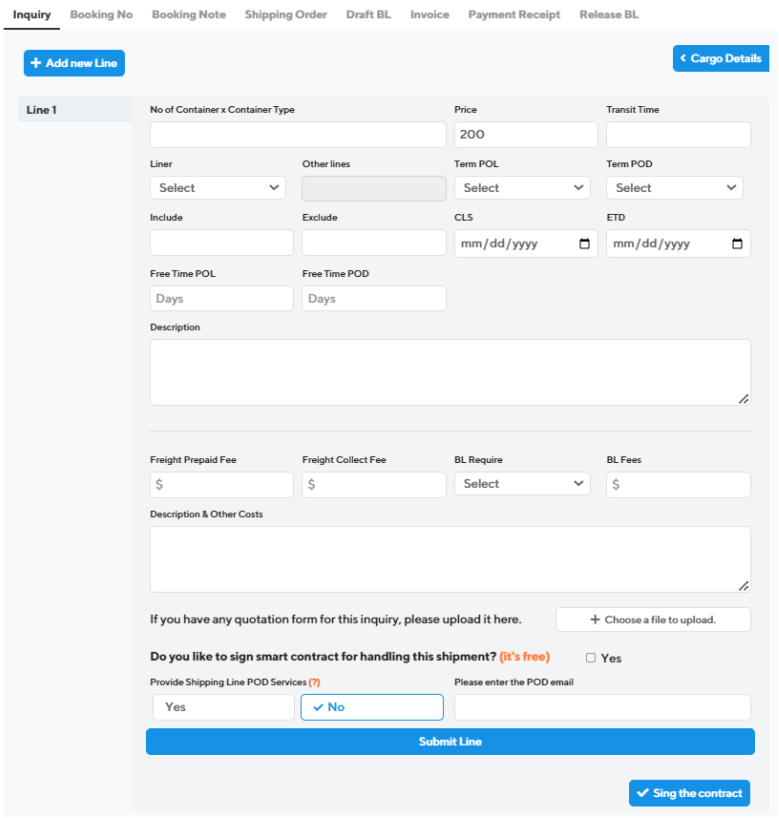
۳-۱-۱- پیاده‌سازی ریزخدمت‌های FCL و Chartering

از آنجایی که پیاده‌سازی این دو زیرخدمت مشابه یکدیگر است، تنها به جزئیات نحوه‌ی پیاده‌سازی زیرخدمت FCL پرداخته خواهد شد. در این برنامه، پیاده‌سازی بلاکچین با زبان پایتون زده شده است. همچنین برای بخش گرافیک نرم‌افزار از کتابخانه QTPython استفاده شده است. جهت پیاده‌سازی بلاکچین از کتابخانه‌های مختلفی استفاده شده که در ادامه به آن پرداخته می‌شود.



شکل ۳ – ۳-۱: نمایی از نرم‌افزار بلاکچین

همانطور که گفته شد، ابتدا باید طرفین قرارداد، قرارداد مورد نظر خود را تایید نهایی کنند. برای انجام این کار، هر کاربر می‌تواند قراردادهای خود را در وب سایت مربوط به پروژه‌ی حمل و نقل مشاهده کند و در صورت تمایل آن را تایید نهایی کند. به عنوان مثال، در تصویر زیر نمونه قراردادی از زیرخدمت FCL آورده شده است که در انتهای آن کاربر می‌تواند با زدن دکمه sign the contract قرارداد مربوطه را تایید نهایی کند. در نتیجه زمانی که تمامی افراد داخل قرارداد، مفاد قرارداد را تایید نهایی کنند، قرارداد مربوطه در برنامه تحت دسکتاپ نمایش داده می‌شود.



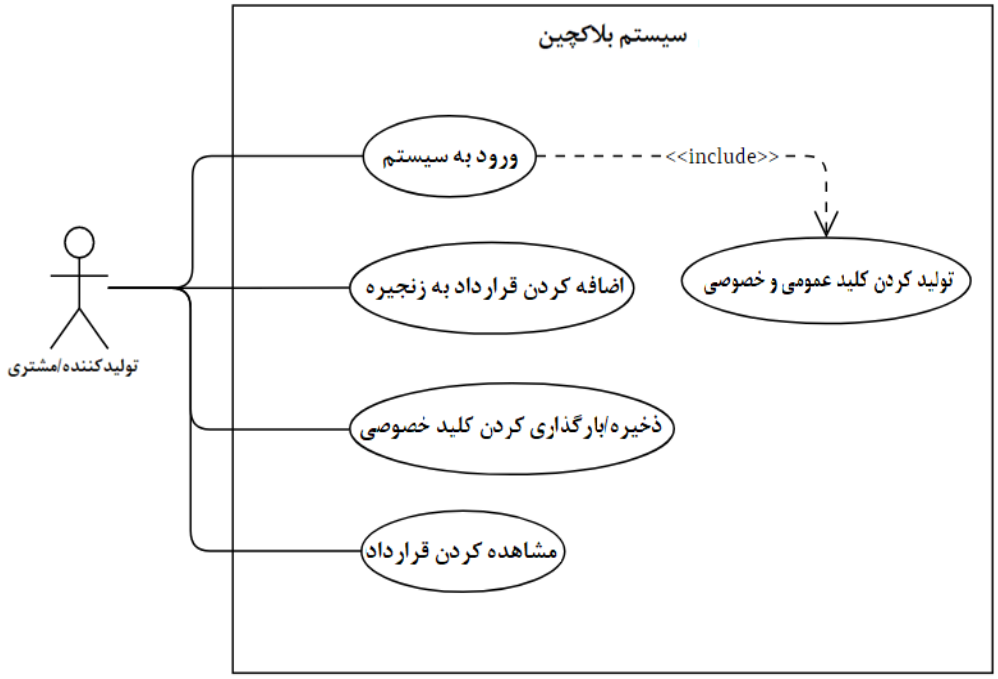
شکل ۳ – ۳-۲: نمونه‌ای از مفاد قرارداد FCL

با توجه به وجود احتمال افزایش کاربران، مشکلاتی در ذخیره‌سازی و انتقال بلاکچین بین کاربران و سرور به وجود می‌آید. دلیل بروز این مشکل این است که افزایش کاربران باعث افزایش حجم قراردادهای عقد شده بین کاربران می‌شود؛ در نتیجه احتمال افزایش حجم زنجیره بلاکچین وجود دارد. در جهت حل این مسئله تصمیم بر این گرفته شد که برای هر قرارداد یک شبکه بلاکچین جداگانه تشکیل شود و در نهایت شبکه مربوطه برای هر قرارداد در سرور ذخیره می‌شود. در نتیجه هر شبکه بلاکچین به تعداد طرفین قرارداد بعلاوه یک بلوک جنسیس، بلوک خواهد داشت. به عنوان مثال، برای قراردادی که طرفین قرارداد آن دو کاربر هستند، تعداد بلوک‌های شبکه بلاکچین مربوط به آن قرارداد سه بلوک خواهد بود. در نتیجه حجم کمی نیاز خواهد بود تا در سرور ذخیره شود. این مسئله میتواند امنیت شبکه بلاکچین را کاهش دهد؛ زیرا در این صورت یک کاربر مخرب می‌تواند با تغییر دادن کل بلوک‌ها (که تعداد آن بسیار کم است) متناسب با خواسته خود، داده‌های ذخیره شده را تغییر دهد.

در ادامه به کمک نمودار مورد کاربرد و نمودار کلاس به جزئیات هر بخش پرداخته شده است. همانطور که گفته شده، از آن‌جایی که پیاده سازی دو زیرخدمت ذکر شده مشابه یکدیگر است و تنها در APIها با هم متفاوت هستند، به توضیحات تنها یکی از این زیرخدمت‌ها، FCL، پرداخته شده است.

- **نمودار مورد کاربرد زیرخدمت FCL**

در پروژه حمل و نقل، بخش بلاکچین یک مورد کاربرد به حساب می‌آید؛ اما برای فهم بهتر جزئیات این بخش، نمودار مورد کاربر آن بصورت جزئی‌تر رسم شده که به شکل زیر می‌باشد. در این نمودار بخش‌های مهم سیستم در قالب نمودار مورد کاربرد بیان شده است.



شکل ۳-۳ – ۳: نمودار مورد کاربرد ریزخدمت FCL

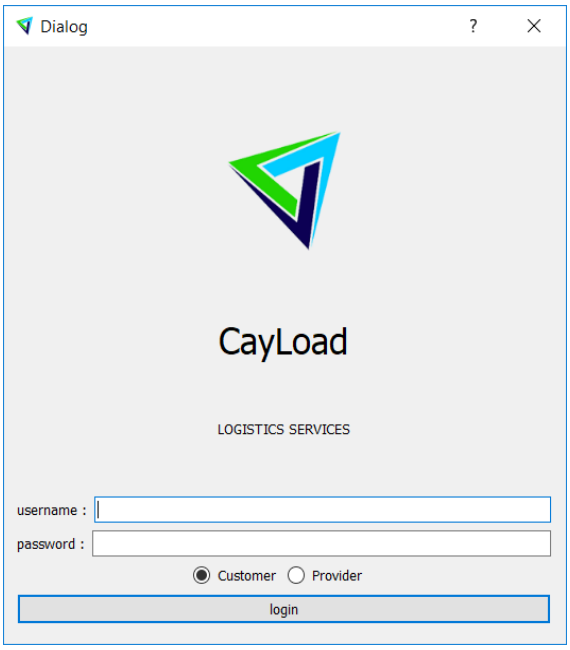
۱. مورد کاربرد ورود به سیستم

در این مورد کاربرد، برای ورود کاربر به نرم افزار از همان مکانیزم و APIهای پیاده سازی شده مربوط به سایت استفاده شده است. در فرایند ورود به سیستم، عملیات تولید کلید نیز صورت می‌گیرد. در طی این فرایند سه حالت برای تولید کلیدها به وجود می‌آید که به شرح زیر می‌باشد:

حالت اول: کاربر برای اولین بار نرم افزار را نصب کرده باشد. در اینصورت فیلد کلید عمومی کاربر در سمت سرور خالی است و تنها نام کاربری و رمز عبور فرد وجود دارد. در اینصورت بعد از ورود موفقیت آمیز کاربر، سیستم کلید عمومی و کلید خصوصی تولید می‌کند و آنها را در سیستم کاربر ذخیره می‌کند؛ همچنین کلید عمومی او به سرور ارسال می‌شود و در آنجا نیز ذخیره می‌شود.

حالت دوم: کاربر نرم افزار را از قبل نصب کرده و مجدد از نرم افزار استفاده می‌کند. در اینصورت کاربر بعد از ورود موفقیت آمیز، از آنجایی که کلید عمومی و خصوصی از قبل تولید شده و در سیستم ذخیره شده است، کلید عمومی و خصوصی جدیدی تولید نمی‌شود و از کلیدهای قبلی استفاده می‌شود.

حالت سوم: کاربر مجبور به نصب مجدد نرم افزار شده است. در اینصورت کاربر بعد از ورود موفقیت آمیز، از آنجایی که کلید عمومی تولید شده او در سرور وجود دارد ولی در سیستم وجود ندارد، مشخص می‌شود کلیدهای عمومی و خصوصی برای او تولید شده است. در نتیجه لازم است کاربر کلید خصوصی خود را بارگذاری کند تا به کمک آن کلید عمومی بازیابی شود و آن را با کلید عمومی موجود در سرور مقایسه شود تا صحت آن تایید شود.



شکل ۳ – ۳-۴: صفحه ورود به نرم افزار

۲. مورد کاربرد تولید کلید عمومی و خصوصی

در این مورد کاربرد به تولید کلید جهت رمزنگاری نامتقارن پرداخته می‌شود. رمزنگاری نامتقارن یک سیستم رمزنگاری است که از دو کلید جهت رمزگذاری و رمزگشایی استفاده می‌کند. جهت تولید کلید عمومی و خصوصی و قابلیت امضا کردن از الگوریتم رمزنگاری RSA موجود در کتابخانه‌ی آماده‌ی cryptography استفاده شده است. برای تولید کلید خصوصی از تابع rsa.generate_private_key() استفاده شده است. یکی از مهمترین پارامترهای این تابع سایز کلید است که معادل ۲۰۴۸ بیت قرار داده شده است. سایز کلید میزان امنیت کلید را مشخص می‌کند که هرچه این سایز بیشتر باشد از امنیت بالاتری برخوردار است. در حال حاضر از آنجایی که روز به روز سیستم‌های قوی‌تری به بازار عرضه می‌شوند در نتیجه حداقل سایز مورد نیاز جهت جعل نشدن کلید در الگوریتم RSA معادل ۲۰۴۸ بیت اعلام شده است. با کلید خصوصی تولید شده، به کمک تابع public_key() کلید عمومی متناظر تولید می‌شود.

بعد از ساخت کلیدها، کلیدهای تولید شده در سیستم کاربر ذخیره می‌شود و کلید عمومی به سرور ارسال می‌شود و در سرور نیز ذخیره می‌شود. از آنجایی که کلیدهای گفته شده از اهمیت زیادی برخوردار هستند،

قبل از ذخیره سازی کلیدها، آنها به کمک کلید متقارن رمزنگاری می‌شوند. برای رمزنگاری متقارن از کتابخانه PBKDF2HMAC و Fernet استفاده شده است.

۳. مورد کاربرد اضافه کردن قرارداد به زنجیره

یکی از بخش‌های مهم بلاکچین در این پروژه، اضافه کردن قرارداد به زنجیره بلوک است. این مورد کاربرد از بخش‌های مختلفی جهت تکمیل فرایند خود استفاده می‌کند که هر یک به شرح زیر است.

- **ایجاد کردن بلوک**

هر بلوک شامل اطلاعات مختلفی است. این اطلاعات به شرح زیر می‌باشد:

- index: شماره بلوک در زنجیره بلوک
- timestamp: زمان ایجاد بلوک
- data: اطلاعات مربوط به قرارداد
- signature: شامل نام، امضا و کلید عمومی کاربر
- previuse_hash: هش اطلاعات بلوک قبلی
- hash: هش اطلاعات بلوک فعلی
- proof_of_work: عدد نانس

برای ایجاد بلوک جدید ابتدا لازم است تمام اطلاعات مربوط به قرارداد از سرور دریافت و به رشته تبدیل شود؛ همچنین لازم است زمان ایجاد بلوک به این رشته اضافه شود. در نتیجه زمان ایجاد بلوک در قالب رشته به اطلاعات قرارداد اضافه می‌شود.

در تکنولوژی بلاکچین، الگوریتم‌های مختلفی برای رسیدن به اجماع استفاده می‌شود. الگوریتم مورد استفاده در این پروژه، الگوریتم اثبات کار[۱] است. این الگوریتم برای رسیدن به اجماع جهت جلوگیری از حملات مربوط به شبکه رایانه‌ای استفاده می‌شود. در این مکانیزم، فرستنده با صرف هزینه پردازش

[۱] Proof of work

محاسبات ریاضیاتی به یک مقدار عددی می‌رسد و گیرنده تنها با کمک آن عدد صحت کار را اثبات می‌کند.

مکانیزم مربوطه با صرف پردازش ریاضیاتی مسئله‌ای را حل میکند. این مسئله با کمک تابع هش SHA256 الگوی مشخص شده را تولید می‌کند. در پروژه الگو داده شده 00 در ابتدا مقدار هش تولید شده است. تا زمانی که هش مورد نظر با این الگو تولید نشده باشد، عملیات تولید هش ادامه خواهد داشت. مقدار ورودی تابع SHA256 رشته داده‌ی قرارداد به همراه زمان و عدد نانس[۲] می‌باشد. در هر بار تولید مجدد هش، مقدار عددی نانس اضافه می‌شود. زمانی که هش با الگوی مورد نظر تولید شد آنگاه مقدار عددی نانس در اطلاعات بلوک با عنوان proof_of_work ذخیره می‌شود. در ابتدا مقدار عددی نانس برابر صفر است.

- **امضا کردن قرارداد**

در ادامه بعد از کامل شدن اطلاعات بلوک و بدست آمدن هش مورد نظر، عملیات امضای کاربر صورت می‌گیرد. کاربر با کلید خصوصی خود هش بدست آمده را امضا می‌کند تا عملیات انجام شده به نام او ثبت شود. با انجام عملیات امضا، امکان انکار کردن از کاربر گرفته می‌شود؛ چرا که تنها کسی که کلید خصوصی او را دارد خودش است؛ بنابراین امضا شدن با کلید خصوصی کاربر به منزله‌ی امضا کردن توسط خود کاربر می‌باشد.

- **جایگزین کردن زنجیره بلوک جدید در سرور**

در ابتدای ایجاد بلوک جدید، آخرین نسخه زنجیره بلوک از سرور دریافت می‌شود. بعد از ایجاد بلوک جدید و امضا شدن آن با کلید خصوصی کاربر، بلوک جدید به زنجیره بلوک دریافت شده اضافه می‌شود.

در ادامه لازم است زنجیره بلوک ایجاد شده جایگزین زنجیره بلوک موجود در سرور شود تا زنجیره بروز شده برای دیگر طرفین قرارداد قابل دسترس باشد. به همین دلیل لازم است قبل ارسال زنجیره بلوک

---

[۲] Nonce

جدید به سرور، شروطی مورد بررسی قرار گیرد؛ چرا که ممکن است در حین ایجاد بلوک جدید، کاربر(ان) دیگر زنجیره بلوک را بروزرسانی کرده باشد و کاربر مذکور از آن مطلع نباشد.

به این منظور در دو مرحله زنجیره بلوک از سرور دریافت می‌شود:

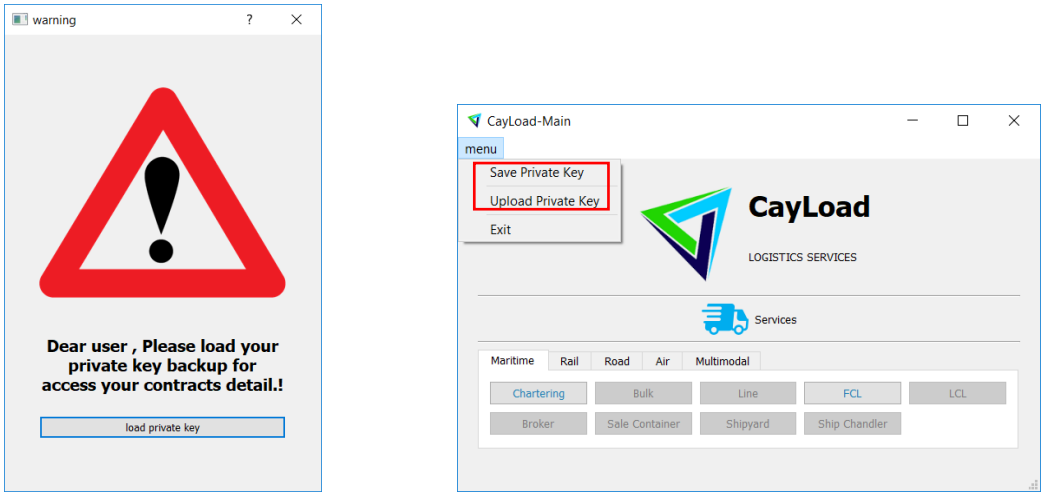
مرحله اول: دریافت آخرین نسخه زنجیره بلوک قبل از ایجاد بلوک جدید (زنجیره بلوک قدیمی)

مرحله دوم: دریافت آخرین نسخه زنجیره بلوک بعد از ایجاد بلوک جدید (زنجیره بلوک جدید)

در این مرحله دو شرط مورد بررسی قرار می‌گیرد. شرط اول بررسی طول زنجیره‌های قدیمی و جدید و شرط دوم بررسی زمان ایجاد آخرین بلوک در زنجیره‌ی قدیمی و جدید است. اگر طول زنجیره قدیمی از طول زنجیره جدید کمتر باشد یا زمان ایجاد آخرین بلوک ایجاد شده در زنجیره بلوک قدیمی جلوتر از زمان ایجاد آخرین بلوک ایجاد شده در زنجیره بلوک جدید باشد، در این صورت زنجیره بلوک طی ایجاد بلوک جدید بروزرسانی شده است. در نتیجه لازم است مجدد بلوک جدید ایجاد شود؛ چرا که بلوک جدید از نظر زمانی باید از آخرین بلوک موجود در زنجیره جلوتر باشد. اگر شرایط فوق برقرار نباشد بدین معناست که زنجیره‌ی ایجاد شده توسط کاربر جدیدترین زنجیره بلوک است و باید جایگزین زنجیره موجود در سرور شود.

۴. مورد کاربرد ذخیره/بارگذاری کلید خصوصی:

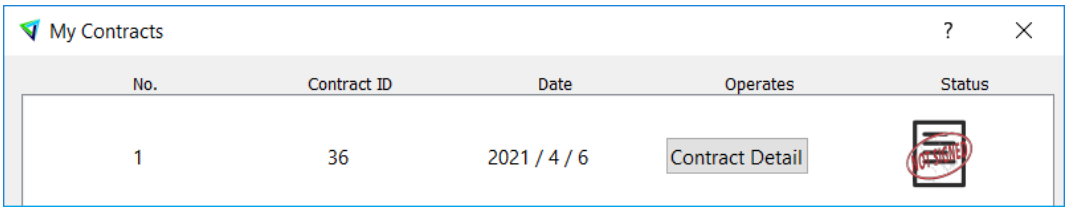
از آنجایی که کلید خصوصی از اهمیت بالایی برخوردار است، امکان ذخیره سازی کلید خصوصی در مسیر دلخواه کاربر پیاده‌سازی شده است. باید در نظر داشته باشیم که بازیابی کلید خصوصی به هیچ روشی امکان پذیر نیست و در صورت پاک شدن، امکان دسترسی به قراردادها از کاربر گرفته می‌شود. در نتیجه در این مورد کاربرد، کاربر می‌تواند کلید خود را ذخیره کند. همچنین همانطور که در بخش ورود به سیستم گفته شد، لازم است در مواقعی کاربر کلید خصوصی خود را بارگذاری کند تا کلید عمومی آن تولید شود. در این مورد کاربرد این امکان به کاربر داده می‌شود.



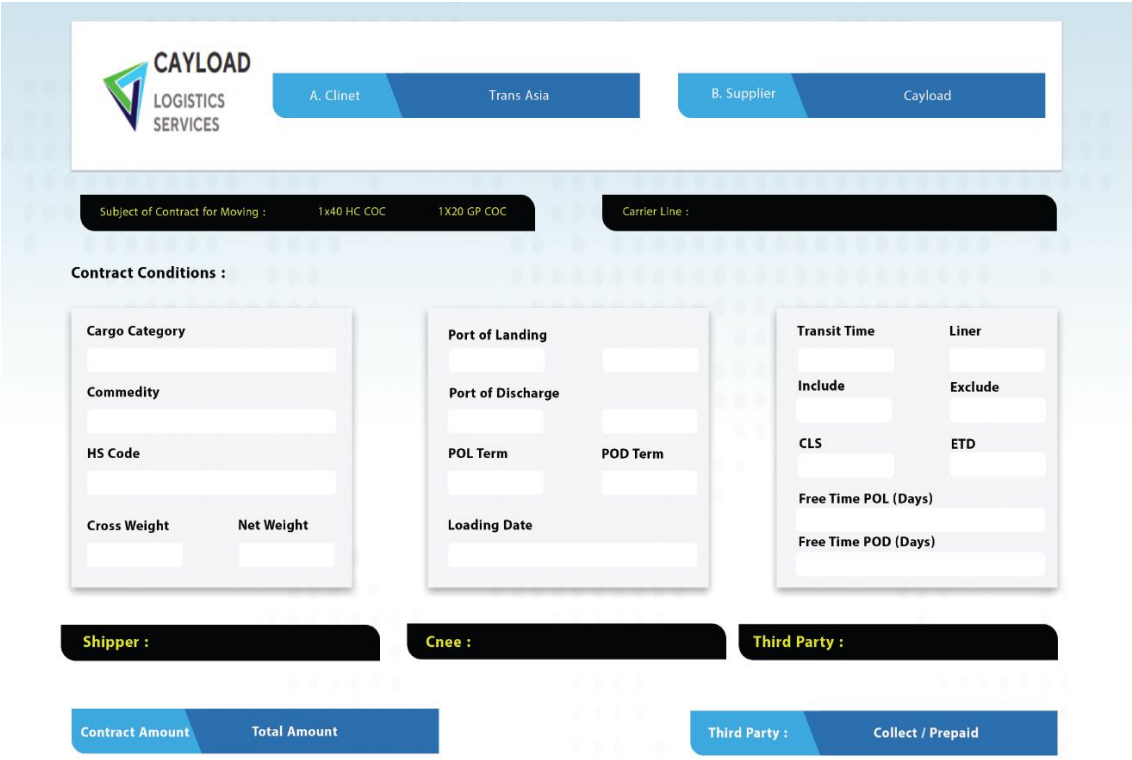
شکل ۳ – ۳-۵: بارگذاری/ذخیره‌سازی کلید خصوصی

۵. مورد کاربرد مشاهده قراردادها:

مشاهده قراردهای نهایی شده توسط طرفین قرارداد برای هر یک از کاربران در نرم افزار مهیا شده است. در صفحه اصلی این نرم افزار تمام خدمات و زیرخدمات قابل مشاهده هستند. همانطور که شکل ۳ - **۳-۱** مشاهده کردید، در حال حاضر تنها بخش‌هایی که دارای بلاکچین هستند فعال شده است. بنابراین لیست قراردادهای نهایی شده در هر خدمت و زیرخدمت در بخش خود قرار گرفته است. به عنوان مثال، شکل ۳ - **۳-۶**، قرارداد نهایی شده در بخش FCL برای کاربر قابل مشاهده است. همچنین کاربران می‌توانند قراردادهای خود را در هر زیرخدمت مشاهده کنند. همچنین این قابلیت به کاربر داده شده تا جزئیات قرارداد خود را در قالب یک فایل PDF مشاهده کند.



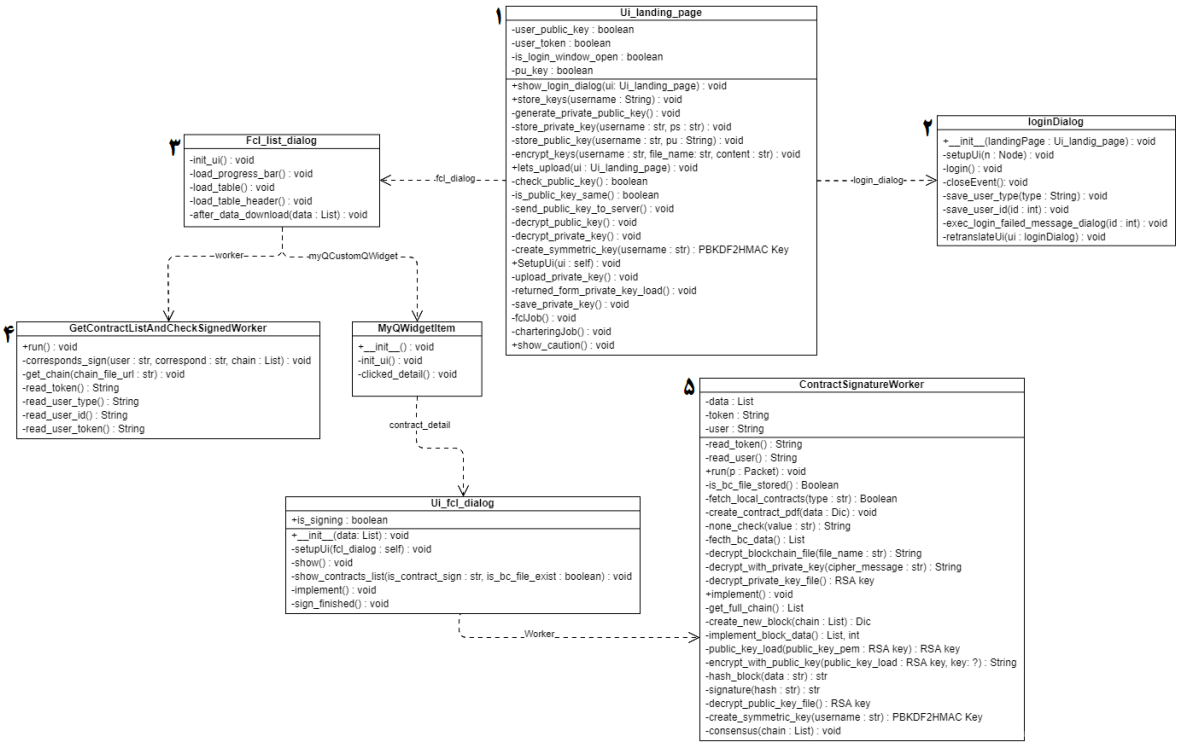
شکل ۳ – ۳-۶: صفحه نمایش لیست قراردادها



شکل ۳ – ۳-۷: نمایی از اطلاعات قرارداد در قالب PDF

- **نمودار کلاس زیرخدمت FCL**

جهت پیاده سازی این نرم افزار، نمودار کلاس آن رسم شده است که در تصویر زیر قابل مشاهده می‌باشد. کلاس شروع کننده کلاس Ui_landing_page می‌باشد که در فایل main.py پیاده سازی شده است. در نمودار زیر، تنها کلاس‌های مربوط به خدمت FCL رسم شده است. سایر خدمات کلاس‌های مشابه و مشترکی دارند و تنها باید API هر یک مطابق خدمت خود جایگزین شود.



شکل ۳ - ۳-۸: نمودار کلاس زیرخدمت FCL

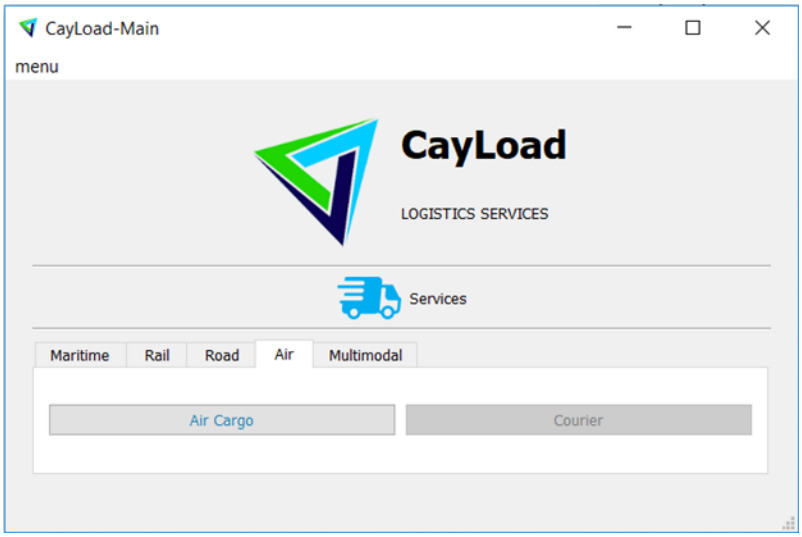
جدول ۳-۳-۲: نگاشت توابع به سند

نام مورد کاربرد	قابلیت	نام کلاس		نام متد
ورود به سیستم	نمایش اولیه صفحه ورود	Ui_landing_page	۱	show_login_dialog()
	فرایند احراز هویت	Ui_loging_dialog	۲	login()
	ذخیره اطلاعات در سیستم میزبان	Ui_loging_dialog	۲	save_user_type() save_user_id()
	بررسی وضعیت کلیدها	Ui_landing_page	۱	lets_upload() check_public_key() is_public_key_same()
تولید کردن کلید عمومی/خصوصی	تولید کلیدها	Ui_landing_page	۱	generate_private_public_key() store_private_key() store_public_key()
	رمزنگاری کلیدها	Ui_landing_page	۱	encrypt_keys() create_symmetric_key()
اضافه کردن قرارداد به زنجیره	ایجاد کردن بلوک	contractSignatureWorker	۵	create_new_block()
	امضا کردن قرارداد	contractSignatureWorker	۵	signature()

consensus()	۵	contractSignatureWorker	جایگزین کردن زنجیره بلوک جدید در سرور	
save_private_key()	۱	Ui_landing_page	ذخیره کردن کلید	ذخیره/بارگذاری کلید خصوصی
upload_private_key()	۱	Ui_landing_page	بارگذاری کردن کلید	
init_ui() load_progress_bar() load_table() load_table_header()	۳	Fcl_list_dialog	ایجاد جدول قراردادها	مشاهده قرارداد
run() corresponds_sign()	۴	GetContractListAndCheckSignedWorker	نمایش لیست قراردادها	
run() create_contract_pdf()	۵	contractSignatureWorker	نمایش به صورت PDF	

۳-۱-۲- پیاده سازی زیرخدمت Air

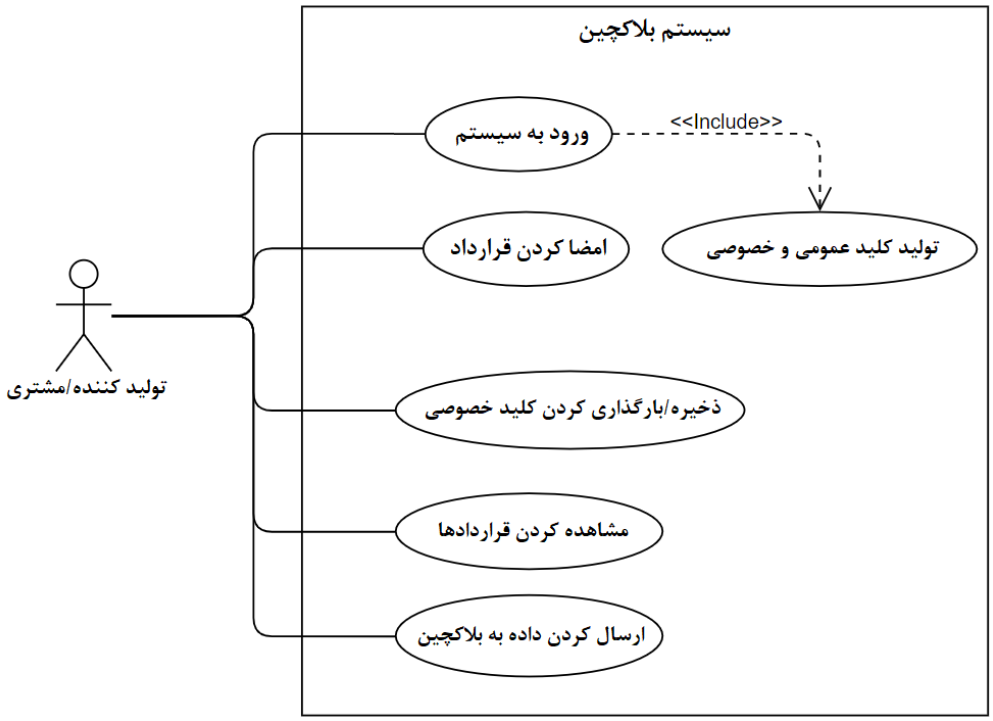
در زیرخدمت Air، برای استفاده از بلاکچین از چارچوب آماده‌ی Fabric Hyperledger استفاده شده است که جزئیات عملکرد آن در بخش پیشینه آورده شده است. در این زیرخدمت، چارچوب ذکر شده جهت ذخیره سازی امن داده‌ها استفاده می‌شود. با توجه به اینکه این چارچوب مشکل مقیاس‌پذیری کمتری نسبت به روش زده شده در دو زیرخدمت دیگر دارد، در نتیجه تمامی قراردادها در یک زنجیره ذخیره می‌شود. در این زیرخدمت، مانند دو زیرخدمت دیگر، از یک نرم افزار یکسان استفاده شده است با این تفاوت که در پیاده‌سازی سرور با یکدیگر متفاوت هستند.



شکل ۳ - ۳-۹: نمایی از نرم افزار – زیرخدمت Air

در ادامه به کمک نمودار مورد کاربرد و نمودار کلاس به جزئیات هر بخش پرداخته شده است.

- **نمودار مورد کاربرد زیرخدمت Air**



شکل۳ – ۳-۱۰: نمودار مورد کاربر زیرخدمت Air

در نمودار مورد کاربر نشان داده شده، تعدادی از مورد کاربردها با ماژول FCL و Chartering یکسان است که شامل موارد کاربرد ورود به سیستم، امضا کردن قرارداد، ذخیره/بارگذاری کردن کلید خصوصی، مشاهده قراردادها و تولید کلید عمومی و خصوصی است. بنابراین از توضیح مجدد آن در این سند خودداری شده است. بنابراین تنها در مورد کاربرد "ارسال کردن داده به بلاکچین" با یکدیگر متفاوت هستند که در ادامه با جزئیات بیشتری به آن پرداخته شده است.

از آنجایی که تنها تعدادی از داده‌ها مهم ضرورت ذخیره سازی در بلاکچین را دارند، در نتیجه باید داده‌های مورد نظر را از بین داده‌های دریافتی از سایت پیش پردازش شود. در نتیجه داده‌ها قبل از ارسال در داخل یک دیکشنری و در فرمت JSON نگهداری می‌شود.

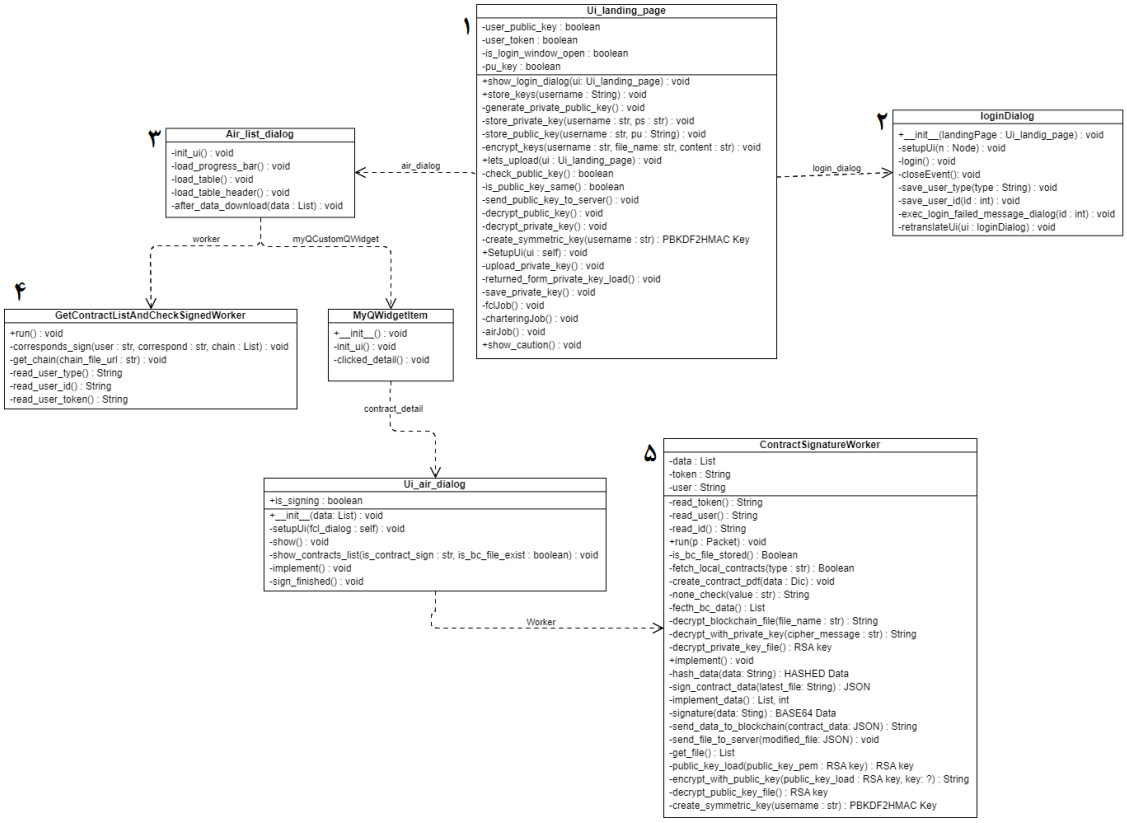
داده‌هایی که در قالب دیکشنری نگهداری می‌شود مطابق زیر است:

- key: آیدی هر قرارداد در بلاکچین (یکتا)
- data: داده قرارداد بدون امضا کاربر
- data_signed: داده‌س امضا شده قرارداد توسط کاربر
- username: نام کاربری
- public_key: کلید عمومی کاربر (کلید متناظر با کلید خصوصی که با آن قرارداد امضا شده است)

داده‌ها بعد از پردازش و آماده شدن، جهت ذخیره سازی در شبکه بلاکچین، به سمت سرور ارسال می‌شوند تا با بررسی آنها داده‌های مربوطه در شبکه ذخیره شوند. مکانیزم اجرایی چارچوب جهت ذخیره‌سازی داده‌ها در بخش پیاده سازی سرور توضیح داده شده است.

- **نمودار کلاس زیرخدمت Air**

جهت پیاده‌سازی این زیرخدمت در نرم افزار، نمودار کلاس آن رسم شده است که در تصویر زیر قابل مشاهده می‌باشد. کلاس شروع کننده کلاس Ui_landing_page می‌باشد که در فایل main.py پیاده‌سازی شده است. در فایل main.py سه زیرخدمت ذکر شده فعال شده است. همانطور که در نمودار مورد کاربرد مشاهده کردید، زیرخدمت Air تنها در یک مورد کاربرد متفاوت است. در نتیجه نمودار کلاس آن با دو زیرخدمت دیگر یکسان است و تنها تعدادی از توابع آن در کلاس contractSignatureWorker تغییر کرده است یا تابعی اضافه شده است. به عبارت دیگر در این کلاس، نحوه پردازش داده و ارسال آن تغییر یافته است. سایر خدمات ارائه شده در این نرم افزار از سمت کاربر با سایر ریزخدمت‌ها یکسان می‌باشد.



شکل۳ - ۳-۱۱: نمودار کلاس زیرخدمت Air

جدول ۳-۳-۳: نگاشت توابع به سند

نام مورد کاربرد	قابلیت	نام کلاس		نام متد
اضافه کردن قرارداد به زنجیره	پردازش کردن داده‌ها	۵	contractSignatureWorker	implement_data()
	امضا کردن قرارداد	۵	contractSignatureWorker	signature()
	ارسال قرارداد به زنجیره	۵	contractSignatureWorker	send_data_to_blockchain()
مشاهده قرارداد	ایجاد جدول قراردادها	۳	Air_list_dialog	init_ui() load_progress_bar() load_table() load_table_header()
	نمایش لیست قراردادها	۴	GetContractListAndCheckSignedWorker	run() corresponds_sign()
	نمایش به صورت PDF	۵	contractSignatureWorker	run() create_contract_pdf()

۳-۲- پیاده‌سازی پروژه – سمت سرور

برای تکمیل فرایند بلاکچین لازم است فرایندهایی سمت سرور اجرا و پردازش شود. در سه زیرخدمت که پیاده سازی بلاکچین انجام شده است، پیاده سازی دو زیرخدمت FCL و Chartering با زیرخدمت Air متفاوت است. در ادامه به جزئیات هر یک پرداخته می‌شود.

۳-۲-۱- پیاده‌سازی زیرخدمت‌های FCL و Chartering

در این دو زیرخدمت پردازش‌های سمت سرور شامل ایجاد فایل اولیه بلاکچین، تایید بلوک جدید و تایید زنجیره بلوک است. همانطور که گفته شد، یکی از قابلیت‌های موجود این نرم افزار مشاهده قراردادها توسط کاربر می‌باشد. شرط نمایش قرارداد در نرم افزار ایجاد فایل اولیه بلاکچین است. به عبارت دیگر، زمانی که برای قرارداد فایل اولیه بلاکچین تولید شد، آنگاه توسط کابر قابل مشاهده خواهد شد؛ در غیر اینصورت کاربر قادر به مشاهده و امضای قراردادی که در سایت ایجاد کرده است در نرم افزار نخواهد بود. به این منظور در مرحله اول، زمانی که قراردادی توسط طرفین قرارداد به تایید نهایی می‌رسد، در سمت سرور فایل اولیه بلاکچین آن قرارداد که شامل بلوک اولیه که به آن بلوک جنسیس گفته می‌شود را تولید خواهد کرد و در مسیر سرویس جاری ذخیره خواهد شد. تایید نهایی کاربر به منزله‌ی عدم تغییر مفاد قراداد خواهد بود؛ به عبارت دیگر، زمانی که طرفین قرارداد، قرارداد مذکور را تایید نهایی می‌کنند، قادر به تغییر محتوای آن نخواهد بود. چرا که بعد از این تایید کاربر قادر به مشاهده قرارداد در نرم افزار و امضای آن خواهد بود.

در ادامه بعد از اینکه کاربر قراردادی را امضا کردن، که امضا کردن قرارداد به معنای ایجاد بلوک جدید و اضافه کردن قرارداد جدید به زنجیره است، لازم است زنجیره بلوک جدید ایجاد شده درستی و صحت آن مورد بررسی قرار گیرد. در مرحله اول باید بلوک جدید اضافه شده مورد بررسی قرار گیرد. در بررسی بلوک جدید دو شرط در آن مورد بررسی قرار میگیرد:

۱- بررسی صحت هش تولید شده بلوک. سرور مجدد با توجه به اطلاعات بلوک، هش آن را محاسبه می‌کند و مقدار آن را با هش تولید شده توسط کاربر مقایسه می‌کند.

۲- بررسی امضای کاربر در بلوک. سرور مجدد با توجه به اطلاعات بلوک، امضای کاربر را مورد صحت سنجی قرار می‌دهد. برای صحت سنجی امضای کاربر از کلید عمومی او استفاده خواهد شد. با کمک کلید عمومی کاربر، اطلاعات امضا شده صحت سنجی می‌شود.

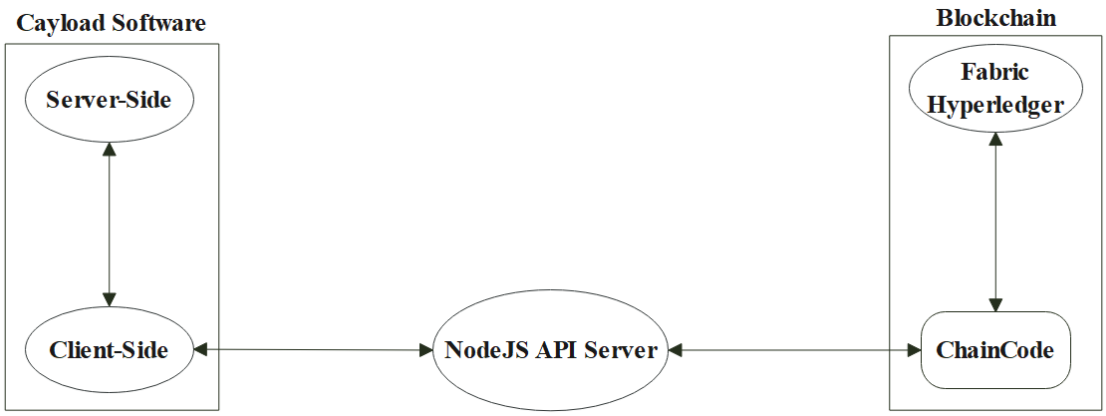
در صورتی که هر دو شرط بالا برقرار باشد آنگاه درستی بلوک تولید شده تایید می‌شود. در مرحله دوم باید زنجیره بلوک صحت سنجی شود. به عبارت دیگر؛ باید ارتباط بین بلوک‌های تولید شده در بلاکچین مورد بررسی قرار گیرد. به این منظور، هش بلوک قبلی که در بلوک فعلی ذخیره شده است با هش خود بلوک قبلی مورد مقایسه قرار می‌گیرد. در صورت یکسان نبودن این دو مقدار می‌توان گفت بلوک قبلی دستکاری شده و در نتیجه زنجیره بلوک موجود درست نمی‌باشد. در غیر اینصورت، صحت زنجیره بلوک تایید خواهد شد.

برای اجرای فرایندهای گفته تنها سه متد در سمت سرور در شرایط لازم اجرا خواهند شد. که عبارتند create_genesis_block()، verify() و verify_sign() می‌باشد.

۳-۲-۲- پیاده‌سازی زیرخدمت Air

در این زیرخدمت برای ذخیره‌سازی داده‌ها از چارچوب Hyperledger Fabric استفاده شده است. این بخش از سیستم، به دلیل کمبود سرور تهیه شده توسط کارفرما، چارچوب مذکور در یک سرور اجرا شده است. لازم به ذکر است با توجه به کارکرد بلاکچین مربوط به چارچوب ذکر شده، می‌توان برنامه مورد نظر را در چندین سرور اجرا کرد که این کار باعث افزایش امنیت زنجیره‌ی بلاکچین می‌شود.

پیاده‌سازی سمت سرور از دو بخش تشکیل شده است. بخش اول مربوط به چارچوب است و بخش دوم مربوط به پیاده‌سازی سرور جهت برقراری ارتباط با فریم. جزئیات هر یک از بخش‌ها را در ادامه خواهیم داشت. بطور کلی معماری نرم افزار به شرح زیر است:



شکل ۳ - ۳-۱۲: شماتیک روابط بین اجزای سازنده سمت سرور

با توجه به مکانیزم عملکرد چارچوب Fabric Hyperledger، برای دسترسی به زنجیره‌ی بلاک‌ها لازم است به کمک یک قرارداد هوشمند نحوه برقراری ارتباط با بلاکچین را مشخص کرد. به عبارت دیگر، هرگونه اقدام برای دسترسی به بلاکچین باید در قرارداد هوشمند برنامه نویسی شود. به عنوان مثال، اگر بخواهیم به یکی از بلوک‌های موجود در بلاکچین دسترسی داشته باشیم، لازم است تابعی را جهت دریافت بلوک مورد نظر در قرارداد هوشمند پیاده‌سازی کرده باشیم. در اصطلاح به این قرارداد هوشمند Chaincode گفته می‌شود.

در سیستم مورد نظر، برای جستوجو کردن بر روی زنجیره، اضافه کردن داده بر روی آن یا دریافت یک قرارداد خاص، سه تابع جداگانه در قرارداد هوشمند پیاده‌سازی شده است که در جدول ۳-۴-۳ آمده است. لازم به ذکر است، با توجه به نحوه عملکرد چارچوب، از آنجایی که این قرارداد هوشمند باید بر روی بلاکچین نصب شود، با هر بار تغییر کد لازم است سیستم بلاکچین مجدد راه‌اندازی شود که این بدین معناست زنجیره جدیدی ایجاد می‌شود. در نتیجه باید در پیاده‌سازی آن دقت لازم را داشت چرا که بعد از نصب، امکان تغییر آن با حفظ اطلاعات موجود در بلاکچین وجود ندارد.

جدول ۳-۳-۴: نگاشت توابع قرارداد هوشمند نصب شده بر روی بلاکچین

API	قابلیت	
InitLedger()	مقداردهی اولیه زنجیره	قرارداد هوشمند (Chaincode)
queryContract()	جستوجوی داده‌ی خاص	
queryAllContract()	جستوجوی تمام داده‌ها	
addContract()	اضافه کردن داده به زنجیره	

از سمت دیگر، با توجه به شکل ۳-۱۲-۳، برای برقراری ارتباط بین بلاکچین و برنامه تحت دسکتاپ لازم است برنامه‌ای به عنوان واسط وجود داشته باشد تا بتواند درخواست‌ها از سمت برنامه تحت دسکتاپ را به قرارداد هوشمند ارسال کند. از آنجایی که قراردادهوشمند تنها با چند زبان محدود برنامه نویسی امکان برقراری ارتباط را دارد، در نتیجه این برنامه به زبان NodeJS که یکی از زبان‌های قابل قبول قرارداد هوشمند است، پیاده‌سازی شده است. این برنامه در قالب REST API زده شده است. برای برقراری ارتباط، سه API زده شده است که در جدول زیر آورده شده است. این برنامه برای برقراری ارتباط با قرارداد هوشمند بازم است اطلاعات شبکه‌ای را که قرارداد هوشمند در آن قرار دارد را داشته باشد و به شبکه متصل شود. به همین منظور تابعی پیاده سازی شده است که به کمک یک کتابخانه، نوشته شده توسط تیم سازنده‌ی چارچوب Fabric Hyperledger، به نام fabric-network می‌توان یک دروازه اینترنت ایجاد کرد که با شبکه مربوطه متصل شد.

جدول ۳-۳-۵: نگاشت توابع مربوط به برنامه واسط

	قابلیت	API/Method	
NodeJS APIs	دریافت تمام قراردادها	query()	GET
	دریافت قرارداد خاص	queryContract()	GET
	اضافه کردن قرارداد	addContract()	POST
	اتصال به شبکه بلاکچین	configNetwork ()	Method

در فرمورک Fabric Hyperledger، برای راه اندازی شبکه بلاکچین از داکر استفاده شده است. در این زیرخدمت، شبکه بلاکچین از سه گره مجازی تشکیل شده که نام آن‌ها Org1، Org2 و Orderer1 است. برای راه اندازی شبکه، دو فایل bash نوشته شده است که به کمک آن فرایند مربوطه، به صورت خودکار و پشت سر هم اجرا می‌شود. در ابتدا فایل startFabric.sh را اجرا می‌کنیم. در این فایل، ابتدا اگر شبکه‌ای موجود باشد آن را غیرفعال می‌کند؛ سپس شبکه جدیدی را ایجاد می‌کند. سپس کانال جدیدی را ایجاد میکند تا به کمک آن گرههای نام برده شده به یکدیگر متصل شوند و با یکدیگر در تعامل باشند. برای ایجاد کانال جدید لازم است که فایل اصلی به نام network.sh اجرا شود؛ که همانطور که گفته شد این کارها بصورت خودکار اجرا می‌شود و لازم به وارد کردن دستورات مورد نظر به صورت دستی نیست. در فایل network.sh توابعی

نوشته شده است که به شرح زیر است. لازم به ذکر است که بعضی از این توابع به یک فایل bash دیگر ارجاع داده شده است. به عبارتی دیگر، فایل network.sh فایل اجرایی اصلی به حساب می‌آید.

جدول ۳-۳-۶: نگاشت توابع در راه‌اندازی شبکه بلاکچین

	قابلیت	توابع	فایل Bash
network.sh	پاک کردن ظرف‌های غیرفعال داکر	clearContainers()	-
	پاک کردن ایمیج‌های اضافی داکر	clearUnwantedImages()	-
	بررسی داشتن پیش‌فرض‌ها	checkPrereqs()	-
	ایجاد گره	createOrgs()	registerEnroll.sh ccp-generate.sh
	ایجاد شبکه جدید	createConsortium()	-
	فعال کردن شبکه	networkUp()	-
	ایجاد کانال	createChannel()	createChannel.sh
	نصب قرارداد هوشمند به کانال	deployCC()	deployCC.sh
	غیرفعال کردن شبکه	networkDown()	-

لازم به ذکر است که شبکه‌ی گفته شده یک شبکه مجازی می‌باشد که امکان دسترسی به آن تنها به افرادی خاص داده شده است و عموم مردم امکان دسترسی به این شبکه را ندارند. برای برقراری ارتباط با قرارداد هوشمند باید احراز هویت صورت بگیرد. در این برنامه تنها دو کاربر مجازی امکان دسترسی به شبکه بلاکچین را دارند (یکی مدیر شبکه و دیگری کاربر عادی) که به کمک یکی از این دو کاربر امکان دسترسی به بلاکچین داده می‌شود. به عبارت دیگر، برنامه واسط برای هر درخواست باید به واسط یکی از این دو کاربر درخواست خود را به قرارداد هوشمند ارسال کند.

جهت مشاهده سایت و دانلود برنامه بلاکچین می‌توانید به دو لینک زیر مراجعه کنید.

آدرس سایت

لینک دانلود برنامه بلاکچین تحت دسکتاپ

۴- ارزیابی

۵- نتیجه‌گیری و کارهای آتی

۶- راهنمای فنی

در این بخش جهت اجرای برنامه راهنمای فنی آن اضافه شده است که با توجه به دو نوع پیاده‌سازی انجام شده، دو راهنمای فنی خواهیم داشت. در ابتدا به بخش فرانت وبسایت که در هر سه زیرخدمت کارهای یکسانی صورت می‌گیرد، پرداخته شده است. سپس به بخش سرور می‌پردازیم. در انتها بخش کاربر را بررسی خواهیم کرد.

۱- راهنمای فنی – فرانت وبسایت Cayload

در ابتدا لازم است قراردادهای منعقد شده توسط طرفین قرارداد تایید نهایی شود. به همین دلیل در وبسایت مورد نظر، در انتهای هر قرارداد دکمه‌ای قرار داده شده است. با کلیک کردن دکمه مورد نظر، تابع handleFinishByRole() اجرا خواهد شد. به کمک این تابع، متغیر finished_by_provider یا finished_by_customer در سمت سرور به حالت True تغییر پیدا می‌کند. عملیات تابع ذکر شده در هر سه زیرخدمت یکسان است و تنها در API ارسال داده متفاوت می‌باشند. آدرس تابع مذکور طبق جدول زیر می‌باشد.

جدول ۶-۱: آدرس فایل فرانت زیرخدمت‌ها

نام زیرخدمت	آدرس فایل
FCL	src/scenes/dashboard/shipping/fcl/info.js
Chartering	src/scenes/dashboard/shipping/chartering/quatation.js
Air	src/scenes/dashboard/ air/info.js

۲- راهنمای فنی – سمت سرور

بخش سرور با زبان پایتون و چارچوب Django زده شده است. از آنجایی که کد این تمام زیرخدمت‌ها یکسان است، تنها به زیرخدمت FCL پرداخته شده است. فایل‌های سمت سرور هر زیرخدمت در جدول زیر آورده شده است.

جدول ۶-۲: آدرس فایل سرور زیرخدمت‌ها

نام زیرخدمت	آدرس فایل
FCL	core/models/services/shipping/fcl.py
Chartering	core/models/services/shipping/chartering.py
Air	

همانطور که در بخش قبل گفته شد، زمانی که هر دو متغیر finished_by_provider و finished_by_customer برابر True باشند، در این صورت فایل بلاکچین آن که شامل بلوک جنسیس است به کمک تابع blockchain_on_finished_contract() تولید خواهد شد.

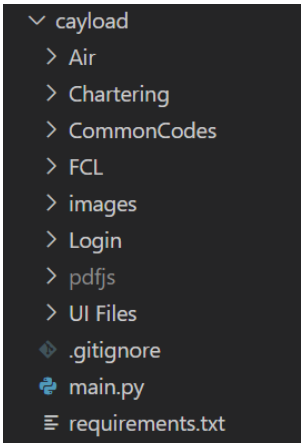
از طرفی دیگر، زمانی که قراردادی توسط کاربر امضا می‌شود، بلوک آن ساخته شده و به سرور ارسال می‌شود. بلوک ارسال شده از سه جهت مورد بررسی قرار می‌گیرد که شامل صحت هش تولید شده، صحت زنجیره تولید شده و صحت امضا می‌شود. در صورتی که هر سه این موارد صحیح باشند، بلوک مورد نظر روی زنجیره قرار میگیرد. در جهت بررسی صحت موارد گفته شده توابع طبق جدول زیر فراخوانی می‌شوند.

جدول ۶-۳: نگاشت توابع سمت سرور

عملیات	تابع
صحت‌سنجی هش	valid_hash()
صحت‌سنجی زنجیره	chain_validity()
صحت‌سنجی امضا	sign_validity()

۳- راهنمای فنی – سمت مشتری

در پیاده‌سازی سمت مشتری، زیرخدمت Air با دو زیرخدمت FCL و Chartering متفاوت هستند که در ادامه به هر یک پرداخته می‌شود. در شکل ۶-۱ زیر پوشه‌بندی پیاده‌سازی سمت سرور نمایش داده شده است. کدهای مربوط به هر زیرخدمت در پوشه‌ی مربوط به خود که هم‌نام با زیرخدمت است، قرار گرفته‌اند. در پوشه CommonCodes و images فایل‌هایی قراردارد که در هرسه زیرخدمت به صورت مشترک استفاده می‌شود. همچنین در پوشه Login فایل‌های مربوط به ورود به نرم افزار قرار گرفته است. پوشه UI Files شامل تمام UIهای نرم افزار می‌باشد که توسط نرم افزار QT Designer طراحی شده‌اند. فایل اجرایی main.py می‌باشد.



شکل ۶-۱: پوشه‌بندی سمت مشتری

در ابتدا برای اجرا برنامه لازم است فرایند زیر اجرا شوند:

۱- نصب پایتون (نسخه ۳ و بالاتر)
۲- نصب کتابخانه virtualenv به کمک ابزار pip
۳- ساخت یک محیط مجازی
۴- نصب کتابخانه‌های مورد نیاز برنامه که در فایل requirements.txt قرار گرفته است. برای نصب این موارد میتوان با اجرا دستور زیر در مسیری که فایل requirments.txt قرار دارد، تمام کتابخانه‌های مورد نیاز را نصب کرد:

```
pip install –r requirments.txt
```

۵- اجرای فایل main.py

هر فایل UI که شامل یک صفحه در نرم افزار است به کمک دستور زیر به فایل پایتون تبدیل می‌شود:

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

بنابراین در هر فایل توابعی برای اجرای UI وجود دارد که مهم‌ترین آن setupUi(self, landing_page) و retranslateUi(self, login_dialog) می‌باشد.

برای آشنایی بیشتر با مسیر اجرایی برنامه، نقشه راه طراحی شده است که به شرح زیر می‌باشد.

از اینجا از وسط دارم راهنمایی مینویسم:

زمانی که کاربر روی دکمه مربوط به هر سرویس میزنه:

سازنده کلاس service_list_dialog فراخوانی میشود که این کلاس در فایل service_contract_list.py قرار دارد. در این کلاس یک شی از کلاس GetContractListAndCheckSignedWorker ساخته میشود و متد run آن اجرا میشود.

در متد run تمام قراردادهای مربوط به کاربر مورد نظر از سرور دریافت می‌شود. قراردادهای دریافتی باید در قالب تعریف شده باشد که به شرح زیر است:

```
{
    "services": [
        /* ONE CONTRACT */
        {
            "id" : "INT",
            "blockchain_chain_file" : "BOOLEAN",
            "number_of_user" : "INT",
            "users" : {
                "user_(ID)" : {
                    "signed" : "BOOLEAN", /* This field is fill by application
*/
                    "id" : "USER_ID"
                }
                /* Based on the number of users, there are user_()
dictionaries */
            },
            "date_created" : {
                "yaer": "",
                "month": "",
                "day": ""
            },
            "data" : {
                /**/
            },
            "signed" : "(FULL OR SIGNED OR NOT)" /* This field is fill by
application */
        }
    ],
    "token" : "USER TOKEN"
}
```

زمانی که قراردادها دریافت شد، به ازای هر قرارداد بررسی میشود که آیا تمام کاربران قرارداد مذکور را تایید نهایی کرده‌اند یا نه. درصورتی که قرارداد توسط تمام طرفین قرارداد تایید نهایی شده باشد باید مقدار متغیر blockchain_chain_file برابر True باشد. اگر مقدار آن False باشد در اینصورت این قرارداد در لیست قراردادهای کاربر نمایش داده نمی‌شود.

درصورتی که قرارداد قابلیت نمایش در لیست کاربر را داشته باشد، با توجه به تعداد منعقد کنندگان قرارداد و افرادی که آن را امضای دیجیتال کرده‌اند، به قرارداد مذکور متغیری تحت عنوان signed اضافه میشود که میتواند سه مقدار FULL، USER و NOT داشته باشد. مقدار FULL برای زمانی است که تمام افراد طرفین قرارداد آن را امضای دیجیتال کرده باشند، مقدار USER زمانی است که تمام افراد امضا نکرده اند ولی کاربر وارد شده به نرم افزار قرارداد مذکور را امضا کرده است و مقدار NOT زمانی است که تمام افراد و خود کاربر امضا نکرده باشند.

بعد از انجام این عملیات، لیست تمام قراردادهایی که قابلیت نمایش در لیست کاربر دارند به عنوان خروجی پاس داده میشود.

بعد از اجرای تابع run، تابع after_data_download اجرا می‌شود. در این متد با توجه به تعداد قراردادها، ردیف در لیست کاربر ایجاد می‌شود. در این تابع، به ازای هر قرارداد یک شی از کلاس MyQWidgetItem ساخته میشود که هر شی شامل اطلاعات قرارداد می‌باشد. در این کلاس برای هر قرارداد دکمه‌ای قرارداده شده است که با کلیک کردن بر روی آن اطلاعات مربوط به هر قرارداد به کلاس Ui_service_dialog پاس داده می‌شود؛ به عبارتی دیگر یک شی از این کلاس ساخته می‌شود و اطلاعات مربوط به قرارداد مذکور به سازنده‌ی آن پاس داده می‌شود. سپس توابعی دیگری از جمله init_ui، load_progress_bar، load_table و load_table_header به ترتیب اجرا می‌شوند.

مراجع

پیوست