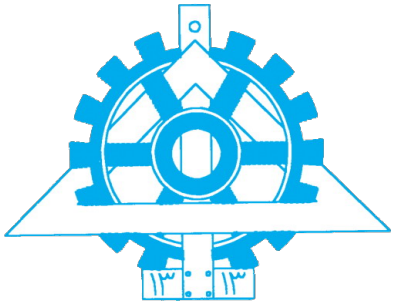


دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر
گروه نرم‌افزار

کاپو، پلتفرم ساخت و انتقال NFT های داده‌ای

پایان‌نامه برای دریافت درجهٔ کارشناسی ارشد در رشتهٔ مهندسی کامپیوتر
گرایش نرم‌افزار

امین بشیری

استاد راهنما

دکتر احسان خامس‌پناه

خرداد ۱۴۰۱


بسم الله الرحمن الرحيم

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر
گروه نرم‌افزار

کاپو، پلتفرم ساخت و انتقال NFT های داده‌ای

پایان‌نامه برای دریافت درجهٔ کارشناسی ارشد در رشتهٔ مهندسی کامپیوتر
گرایش نرم‌افزار

امین بشیری

استاد راهنما

دکتر احسان خامس‌پناه

خرداد ۱۴۰۱

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر

گواهی دفاع از پایان‌نامه کارشناسی ارشد

هیأت داوران پایان‌نامهٔ کارشناسی ارشد آقای / خانم امین بشیری به شمارهٔ دانشجویی ۸۱۰۱۹۶۴۲۵ در رشتهٔ مهندسی کامپیوتر - گرایش نرم‌افزار را در تاریخ ..... با عنوان «کاپو، پلتفرم ساخت و انتقال NFT های داده‌ای »

	به عدد	به حروف
با نمرهٔ نهایی		

و درجهٔ [ ] ارزیابی کرد.

ردیف	مشخصات هیأت داوران	نام و نام خانوادگی	مرتبهٔ دانشگاهی	دانشگاه یا مؤسسه	امضا
۱	استاد راهنما	دکتر احسان خامس‌پناه	استاد	دانشگاه تهران	
۲	استاد داور داخلی	دکتر داور داخلی	دانشیار	دانشگاه تهران	
۳	استاد مدعو	دکتر داور خارجی	دانشیار	دانشگاه داور خارجی	
۴	نمایندهٔ تحصیلات تکمیلی دانشکده	دکتر نماینده	دانشیار	دانشگاه تهران	

نام و نام خانوادگی معاون تحصیلات تکمیلی و پژوهشی دانشکده / گروه:

تاریخ و امضا:

نام و نام خانوادگی معاون آموزشی و تحصیلات تکمیلی پردیس دانشکده‌های فنی:

تاریخ و امضا:

تعهدنامهٔ اصالت اثر

باسمه تعالی

اینجانب امین بشیری تأیید می‌کنم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب است و به دستاوردهای پژوهشی دیگران که در این نوشته از آن‌ها استفاده شده است مطابق مقررات ارجاع گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتری ارائه نشده است.

نام و نام خانوادگی دانشجو: امین بشیری

تاریخ و امضای دانشجو:

چکیده

«اینترنت غیرمتمرکز[1]» یا Web۳ به عنوان مهمترین تغییر بعد از به وجود آمدن اینترنت[2] در نظر گرفته می‌شود. با به وجود آمدن «رمزارزها[3]» و «الگوریتم‌های اجماع[4]» کارآمد، و فراهم شدن زمینه اجرای برنامه‌ها و انجام تراکنش‌های مالی به صورت غیر متمرکز، عصر اینتزنت غیرمتمرکز فرا رسیده است.

در این میان یکی از اصلی‌ترین مزایای «برنامه‌های غیرمتمرکز[5]» مالکیت واقعی دارایی است، به این معنی که یک شخص یا یک نهاد نمی‌تواند دارایی‌های کس دیگری را مسدود یا مصادره کند. از طرفی مالیکت‌های معنوی هم به صورت واضح و شفاف می‌توانند مشخص شوند. برای مثال یک هنرمند به وضوح صاحب اثرش است و هرچند که دیگر افراد می‌توانند اثر او را کپی کنند اما همیشه مشخص است که صاحب اصلی اثر کیست.

به این ترتیب «توکن‌های تعویض ناپذیر[6]» با قابلیت‌های مالکیت بسیار زیادی که فراهم می‌کنند مورد استقبال فراوان مردم واقع شدند. قابلیت‌هایی مانند ساخت، نگهداری، فروش و انتقال فوق‌العاده راحت و سریع نیز در این سرعت فراگیری تاثیر بسزایی داشته‌اند. برای بازاری به این تازگی و وسعت، تکنولوژی‌ها، استانداردها و پلتفرم‌های زیادی ساخته شده‌اند و همچنان نیز در حال توسعه هستند.

در این پروژه سعی بر ساخت پلتفرمی داریم که هر شخص یا شرکتی بتواند با عضویت در آن، به آسان‌ترین روش ممکن توکن‌های تعویض ناپذیر بسازد و به دیگران انتقال دهد. کاربردهای این پلتفرم ساده بی‌شمار است. دارایی‌هایی مانند بلیت سینما، ژتون‌های غذا، وقت گرفتن از دکتر، قراردادها و ... همه می‌توانند به آسانی در این پلتفرم به توکن تعویض ناپذیر تبدیل شوند، به دیگران انتقال یابند و در بازار خرید و فروش شوند.

اگرچه کاربردهای فراوانی برای این پلفرم می‌توان در نظر داشت اما همچنان هدف اصلی از انجام این پروژه آشنایی با نحوه ساخت، تست و دیپلوی یک اسمارت کانترکت، ساخت فرانت‌اند، اتصال آن به اسمارت کانترکت و همچنین شناخت استانداردهای معروف قراردادهای توکن‌های تعویض ناپذیر مانند ERC721 و ERC1155 است.

لازم به ذکر است که تمامی کدهای کاپو به صورت متن‌باز در گیت‌هاب قابل دسترس برای عموم هستند.

---

[1] Decentralized Web
[2] Word Wide Web
[3] CryptoCurrencies
[4] Consensus Algorithms
[5] Dapps
[6] Non-fungible tokens

واژگان کلیدی کاپو، توکن، داده، سالیدیتی، اسمارت کانترکت، قرارداد هوشمند، توکن غیرقابل تعویض، cappu، solidity، ERC721، ERC1155، non-fungible

فهرست مطالب

فصل ۱

مقدمه و بیان مسئله

۱.۱ مقدمه

در یک دهه اخیر محبوبیت رمزارزها در میان مردم به شدت افزایش داشته است. رمزارزها توکن هایی تعویض پذیر هستند به این معنی که تفاوتی میان دو توکن یک رمزارز وجود ندارد، مانند پول فیزیکی[1] که ارزش یک هزار تومانی با یک هزار تومانی دیگر تفاوتی ندارد.

اما در دنیای واقعی تنها مالکیت پول نیست که اهمیت دارد، بلکه یک فرد میتواند خودرو، خانه، بلیت هواپیما و دیگر دارایی‌هایی داشته باشد که یکتا هستند و با هیچ دارایی دیگری دقیقا یکسان نیستند. مثلا یک بلیت هواپیما برای تاریخ و ساعتی خاص برای شماره پروازی خاص از یک مبدا مشخص به یک مقصد مشخص است و شماره صندلی یکتایی نیز دارد. پس هیچ دو بلیت هواپیمایی دقیقا یکسان نیستند، بر خلاف دو بیتکوین که کاملا یکسان هستند، ارزش برابری دارند، و تعویض پذیر هستند.

کاربردهای توکن‌های تعویض ناپذیر بیشمار است و در حال حاضر فقط قسمت اندکی از کاربردهایی که میتوانند داشته باشند را پاسخ گفته‌اند. در این پروژه یک پلتفرم[2] می‌سازیم که ساخت و انتقال توکن‌های تعویض ناپذیر را برای عموم در دسترس‌تر و آسان‌تر می‌کند. همچنین یکی از اهدف انجام این پروژه آشنایی با تکنولوژی‌ها، استانداردها و فرایندهای توسعه این توکن‌هاست.

---

[1] Fiat Money
[2] Platforms

۲.۱ شرح مسئله و روش انجام آن

پروژه تعریف شده توسعه یک پلتفرم برای ساخت[۳] و انتقال توکن‌های تعویض ناپذیر به آسان‌ترین روش ممکن است، به نحوی که برای هر کسی به راحتی در دسترس باشد. نکته‌ی قابل توجه این است که در مسیر انجام این پروژه با تکنولوژی‌های موجود در این زمینه، فریمورک‌ها، استانداردها و فرایند تست و دیپلوی آشنا شویم.

برای انجام این مراحل در قدم اول نحوه توسعه اپلیکیشن‌های غیرمتمرکز و برتری‌های نوشتن پروژه به صورت متن‌باز ذکر می‌شود، سپس فریمورک‌ها و ابزارهایی که برای ساخت یک اپلیکیشن غیرمتمرکز به توسعه دهنده کمک می‌کنند معرفی می‌شوند و نحوه استفاده از آن‌ها شرح داده می‌شود.

سپس فرایند توسعه آغاز می‌شود، استانداردهای موجود برای نوشتن یک قرارداد برای توکن‌های تعویض ناپذیر شرح داده می‌شود و کاپو تا جای ممکن مطابق آن‌ها توسعه می‌یابد. برای قرارداد هوشمند نوشته شده تست می‌نویسیم و آن را روی شبکه تستی[۴] انتشار می‌دهیم. در گام بعد برای پلتفرم، فرانت‌اند ساده‌ای نوشته می‌شود که با قرارداد هوشمند و همچنین کیف پول دیجیتال کاربر ارتباط برقرار می‌کند و سپس به کمک صفحات گیت‌هاب[۵] دیپلوی می‌شود تا در دسترس عموم کاربرها قرار بگیرد.

برای داکرایز[۶] کردن تست‌های قرارداد هوشمند[۷] یک ایمیج داکر[۸] ترافل[۹] نوشته می‌شود. در قدم بعد هر دو بخش فرانت و قرارداد هوشمند داکرایز می‌شوند و فرایند اجرای تست‌های قرارداد هوشمند و دیپلوی شدن فرانت به صورت خودکار به کمک پایپلاین‌های گیت‌هاب پیاده‌سازی می‌شود.

۳.۱ اهداف کلی تحقیق

اهداف این تحقیق را می‌توان به دو دسته تقسیم‌بندی نمود.

[3] Mint
[4] Testnets
[5] Github Pages
[6] Dockerize
[7] Smart Contracts
[8] Docker Images
[9] Truffle

۱.۳.۱ گسترش کاربردهای توکن‌های تعویض ناپذیر

این توکن‌ها در همین مدت کوتاهی که به وجود آمده‌اند کاربردهای فراوانی را پوشش داده‌اند. اما همچنان قسمت بزرگی از این کاربردها صرفا ثبت مالکیت آثار هنری دیجیتال است. درحالی که توکن‌های داده‌ای می‌توانند وسعت بسیار عظیم‌تری از کاربردها را پوشش دهند. از کاربردهای روزانه مانند بلیت سینما و هواپیما، تا مالکیت هر نوع دارایی واقعی یا مجازی.

با توجه به نحوه کار اکثر قراردادهای توکن‌های تعویض ناپذیر، معمولا فقط مالک قرارداد می‌تواند توکن ایجاد کند، یا در قرارداد برای ایجاد توکن شرط‌هایی مانند حداکثر تعداد ممکن گذاشته می‌شود. این موضوع به این معنی است که اگر شخصی بخواهد خودش توکن‌هایی ایجاد کند و به دیگران انتقال دهد احتمالا مجبور است که قراردادهوشمند خودش را بنویسد و دیپلوی کند. این فرآیند نیاز به دانش فنی، آشنایی کامل با این زمینه و پرداخت هزینه‌های دیپلوی قرارداد روی شبکه بلاکچین دارد.

کاپو به هر آدرسی اجازه می‌دهد که به راحت‌ترین حالت ممکن و به هر تعداد که مورد نیاز است توکن تعویض ناپذیر روی این قرارداد ایجاد کند. به این ترتیب استفاده از کاپو برای عموم مردم آسان‌تر، ارزان‌تر و در دسترس‌تر است.

۲.۳.۱ یادگیری

هدف دیگر انجام این پروژه یادگیری است. با توجه به رشد سریع و تازگی استفاده از تکنولوژی‌های بلاکچین و توکن‌های تعویض ناپذیر، با وجود تلاش برای ایجاد منابع یادگیری مناسب همچنان فضاهای خالی، کمبودها و نیازمندی‌هایی وجود دارد که باید پاسخ گفته شوند. در طی انجام این پروژه با ابزارها، کتابخانه‌ها، فریمورک‌ها و استانداردهای نوشتن قراردادهای هوشمند آشنا می‌شویم، می‌آموزیم که هر یک چطور کار میکنند و چگونه می‌توانند به توسعه دهنده کمک کنند.

۴.۱ ساختار پایان‌نامه

پس از این مقدمه، در فصل ۲ مفاهیم اولیه توسعه اپلیکیشن بر بستر بلاک‌چین، کاربردها، مفاهیم و استانداردها توضیح داده می‌شود. در فصل ۳ ابزارهای توسعه قراردادهای هوشمند معرفی می‌شوند، مزایا و معایب هر یک بیان می‌شود و نحوه استفاده از آن‌ها توضیح داده می‌شود. در فصل ۴ روند پیاده‌سازی شرح داده می‌شود. بررسی می‌شود که در هر مرحله از پیاده‌سازی چه کارهایی به چه ترتیبی انجام شده است. در فصل پنجم نیز نتایج توضیح داده می‌شوند و جمع‌بندی صورت میگیرد.

فصل ۲

مفاهیم اولیه و پیش‌زمینه

۱.۲ دلایل و برتری‌های متن‌باز بودن قراردادهای هوشمند

دلایل زیادی برای متن‌باز نوشتن قراردادهای هوشمند وجود دارد، در ادامه تعدادی از این دلایل توضیح داده می‌شود.

دلیل اول، بلاک‌چین‌ها محرمانگی[1] ندارند، همه‌ی نودهای شبکه برای اجرای کد قرارداد هوشمند باید حداقل به بایت‌کدها[2] قرارداد هوشمند دسترسی داشته باشند و این بایت‌کدها در کاوشگرهای بلاکچین نیز وجود دارند، همچنین دیکامپایلر[3]هایی وجود دارند که از بایت‌کدهای قرارداد هوشمند کد سالیدیتی آن را به دست می‌آورند. پس در نتیجه تلاش برای مخفی کردن کدهای قرارداد هوشمند بیهوده خواهد بود.

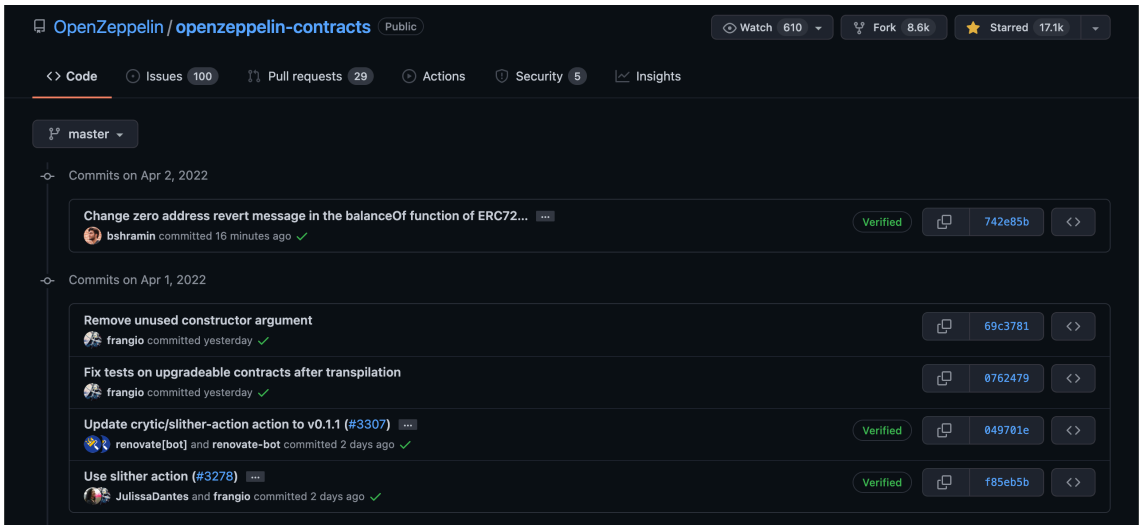
دلیل دوم، اصلی‌ترین مذیت اپلیکیشن‌های غیرمتمرکز نسبت به اپلیکیشن‌های متمرکز عدم نیاز به اعتماد است، کاربرها می‌توانند کدهای قرارداد هوشمند را بخوانند و به کد نوشته شده اعتماد کنند،در حالی که اگر کد برنامه برای همه کاربران قابل مشاهده نباشد کاربرها باید به سازندگان آن برنامه اعتماد کنند.

دلیل سوم، دیپلوی کردن قراردادهای هوشمند معمولاآسان نیست وسرعت تغییرات پایین‌تر از اپلیکیشن‌های متمرکز هست، پس امکان این که با پیدا شدن هر مشکل بتوان به سرعت آن را درست کرد کمتر وجود دارد و مسئله امنیت بسیار اهمیت دارد. متن‌باز نوشتن قرارداد هوشمند باعث می‌شود چشم‌های بیشتری کدهای قرارداد

[1] Confidentiality
[2] bytecodes
[3]

را بخوانند و مشکلات احتمالی سریعتر مشخص و رفع شوند. تعداد زیادی از این پروژه‌ها از همان روز اول قرارداد هوشمند را به صورت متن‌باز توسعه می‌دهند، بعضی نیز ترجیح میدهند که پروژه به مرحله‌ای از توسعه برسد و سپس آن را متن‌باز میکنند.

ر این حوضه سرعت پیشرفت و توسعه به دلیل متن باز بودن به شدت بالاست به نحوی که در طی اجرای این پروژه مرج ریکوئستی روی کتابخانه OpenZeppelin زده شد که در همان روز مرج شد. این موضوع علاوه بر این که نشان‌دهنده سرعت پیشرفت بسیار بالاست، این موضوع را نیز نشان میدهد که در یک جامعه متن‌باز هر توسعه دهنده می‌تواند به پیشرفت جامعه به هر شکلی که می‌تواند کمک کند، اشکالاتی که مشاهده میکند را گزارش دهد یا تصحیح کند.



شکل ۱.۲: در طی انجام پروژه مرج ریکوئستی روی OpenZeppelin باز شد که در همان روز مرج شد.

۲.۲ آشنایی با مفهوم توکن تعویض ناپذیر

شروع رمزارزها با توکن‌های تعویض پذیر بود، مفهوم تعویض پذیری به این معنی است که یک توکن با توکن دیگر تفاوتی ندارد و با جابه‌جا شدن آن‌ها تغییری ایجاد نمی‌شود. برای مثال یک بیت‌کوین با یک بیت‌کوین دیگر هیچ تفاوتی ندارد.

اما توکن‌های تعویض ناپذیر اینگونه نیستند، هر یک منحصر به فرد است و جابه‌جا کردن آن‌ها با یکدیگر

تغییر ایجاد میکند، در دنیای واقعی خانه می‌تواند مثال خوبی از یک دارایی تعویض ناپذیر باشد، هیچ دو خانه‌ای دقیقا شبیه به هم، در یک مکان، در طبقه یکسان و دارای پلاک مشترک نیستند.

پس مثلا به عنوان یک کاربرد، شهرداری می‌تواند یک قرارداد هوشمند ایجاد کند و به هر خانه یک توکن NFT اختصاص دهد. به این صورت صاحب خانه به جای سند یک توکن NFT دارد که مشخص می‌کند که دارایی مطعلق به اوست، و فروش خانه به راحتی انتقال آن NFT به شخص دیگری است.

از نظر فنی هر توکن به این صورت یکتاست که یک TokenId یکتا در قراردادش دارد و هر قرارداد هم دارای یک آدرس یکتا در شبکه بلاکچین است. پس ترکیب Address Contract و TokenId باعث می‌شود که هر توکن یکتا باشد.

۳.۲ کاربردها، حال و آینده

کاربرد هاNFT تا به حال در دو دسته خلاصه می‌شود. دسته اول به عنوان صاحب یک اثر دیجیتال، مانند یک تصویر یا یک موسیقی. دسته دوم به عنوان یک جواز یا بلیت برای ورود به جایی یا دریافت چیزی، برای مثال همایشی برگذار می‌شود که فقط دارندگان هایNFT یک قرارداد هوشمند می‌توانند به آن وارد شوند.

معروف‌ترین پلتفرم معاملاتی این توکن‌ها OpenSea است که می‌توان در آن توکن‌های موجود را مشاهده کرد و یک توکن را توسط مزایده خرید یا به فروش گذاشت. OpenSea در حال حاضر از قراردادهای شبکه‌های اتریوم و سولانا پشتیبانی میکند. دیگر شبکه‌ها نیز معمولا پلتفرم‌های خود را دارند، مانند شبکه Atom که در آن از پلتفرم Stargaze برای معامله هاNFT استفاده می‌شود.

کاربردهای NFT ها در آینده می‌تواند بسیار وسیع باشد. دارایی‌های فیزیکی دنیای واقعی، بلیت‌های ورود به یک مکان یا یک همایش، دارایی‌های دنیای مجازی مانند یک موسیقی یا آیتمی در یک بازی و حتی دامنه‌های اینترنتی همه می‌توانند به NFT تبدیل شوند. مزایای تبدیل این موارد به NFT قابلیت نگهداری آسان‌تر، قابلیت فروش و انتقال راحت‌تر، امنیت بیشتر، آزادی در تراکنش‌ها و آشکار بودن مالکیت دارایی بر همگان است.

۴.۲ قراردادهای هوشمند و استانداردسازی

اکثر قراردادهای هوشمند قابلیت‌هایی مشابه با یکدیگر دارند، برای مثال گروهی از قراردادهای هوشمند توکن‌های تعویض پذیر دارند و گروه توکن‌های تعویض ناپذیر. از طرفی اپلیکیشن‌هایی مانند کیف‌پول‌های دیجیتال، پلتفرم‌های معاملاتی و صرافی‌های نیاز دارند که بتوانند دارایی‌های کاربر اعم از توکن‌های تعویض پذیر و تعویض ناپذیر را ببینند، به همین دلیل باید نحوه صحبت کردن با قراردادهای هوشمند را بدانند.

برای ساده‌تر کردن این فرایند و همسان‌سازی اینترفیس این قراردادهای هوشمند استانداردهایی تعریف شده است که با استفاده از این استانداردها هم فرایند توسعه اسمارت کانترکت آسان‌تر خواهد شد و هم ارتباط میان قراردادهوشمند و اپلیکیشن‌های دیگر مانند کیف‌پول‌ها، پلتفرم‌های معاملاتی و ... آسان‌تر خواهد شد.

از نمونه‌های معروف این استانداردها ERC20 برای قراردادهایی با توکن‌های تعویض پذیر و ERC721 برای قراردادهایی با توکن‌های تعویض ناپذیر است. در این پروژه از استاندارد ERC721 استفاده می‌شود اما در مورد ERC1155 هم مطالعه شده و توضیح داده می‌شود، به طور خلاصه ERC1155 قابلیت‌های بیشتری از ERC721 دارد و یک قرارداد با این استاندارد می‌تواند هم توکن‌های تعویض پذیر و هم تعویض ناپذیر داشته باشد.

برای استفاده از این استانداردها از پکیج‌های متن بازی استفاده می‌شود که این استانداردها را پیاده‌سازی کرده‌اند و از آن‌ها در قراردادی که نوشته می‌شود ارث‌بری می‌شود، یکی از بهترین پیاده‌سازی‌های این استانداردهای توسط اپن‌زپلین[۴] انجام شده است که در این پروژه نیز از همین پیاده‌سازی استفاده می‌شود.

۱.۴.۲ استاندارد ERC20

این استاندارد مناسب توکن‌های تعویض پذیر است. اینترفیسی تعریف می‌کند که نیازهای قراردادهایی با توکن‌های تعویض پذیر را برطرف کند و نحوه تعامل برقرار کردن با آن‌ها را یکسان گرداند. در این استاندارد فقط می‌توان یک نوع توکن تعویض پذیر به تعداد دلخواه داشت. این استاندارد متدهایی برای تعریف حداکثر تعداد توکن‌های موجود، گرفتن موجودی یک آدرس، و انتقال توکن‌ها دارد. توضیحات دقیق‌تر در مورد این استاندارد را می‌توان در وبسایت اتریوم [۵] یا اپن‌زپلین [۶] مشاهده کرد.

[4] OpenZeppelin
[5] https://ethereum.org/en/developers/docs/standards/tokens/erc-20
[6] https://docs.openzeppelin.com/contracts/4.x/api/token/erc20

۲.۴.۲ استاندارد ERC721

استفاده از استاندارد ERC721 برای توکن‌های تعویض ناپذیر بسیار مرسوم است. در این استاندارد متدها و ایونت‌هایی برای یکسان سازی اینترفیس قراردادهای دارای توکن‌های تعویض ناپذیر تعریف شده است. در این نوع قراردادها میتوان به تعداد دلخواه توکن‌های متفاوت با یکدیگر داشت، هر توکن یک آیدی یکتا دارد که می‌تواند به صورت ترتیبی یا غیر ترتیبی ایجاد شود.

همچنین متدی وجود دارد که میتواند آیدی یک توکن را به آدرسی تبدیل کند که اطلاعات آن توکن در آنجا موجود است. کاربرها می‌توانند توکن‌هایی که دارند را مشاهده کنند، به یکدیگر ارسال کنند یا به آدرس دیگری وکالت بدهند که توکن‌ها را به شخص دیگری ارسال کند.

تنها قابلیتی که به طور مشخص در این قرارداد معین نشده است که چگونه باید انجام شود قابلیت ساخت توکن‌ها است. اکثر قراردادهای هوشمندی که توکن‌های تعویض ناپذیر دارند به کاربران اجازه ساخت توکن‌ها را نمی‌دهند و ساخت توکن‌ها فقط به آدرس صاحب قرارداد محدود می‌شود. اما در کاپو اینگونه نیست و هرکسی می‌تواند برای خودش توکن بسازد.

اطلاعات دقیق‌تر در مورد این استاندارد را نیز می‌توان در وبسایت اتریوم [۷] یا اپن‌زپلین [۸] مشاهده کرد.

۳.۴.۲ استاندارد ERC1155

تا اینجا با معروف‌ترین استانداردهای موجود برای قراردادهایی که توکن‌های تعویض پذیر یا تعویض ناپذیر دارند آشنا شدیم. اما همچنان نیازمندی‌هایی وجود دارند که توسط هیچ‌یک از این استانداردها برطرف نمی‌شوند. نیازمندی‌هایی مانند:

- داشتن توکن‌های NFT با تعداد محدود به جای فقط یکی.
- داشتن همزمان چندین نوع توکن مختلف در یک قرارداد.
- انتقال همزمان چند توکن از انواع مختلف از کاربری به کاربر دیگر.

[7] https://ethereum.org/en/developers/docs/standards/tokens/erc-721
[8] https://docs.openzeppelin.com/contracts/4.x/api/token/erc721

یک مثال از کاربردی که به این قابلیت‌ها نیاز دارد می‌تواند یک بازی مثل مونوپولی باشد که در آن هر کاربر مقداری پول دارد که در واقع یک توکن تعویض پذیر هست، به عنوان دارایی چند خانه دارد که به عنوان توکن‌های تعویض ناپذیری هستند که از هرکدام فقط یکی وجود دارد و ممکن است چند کارت خروج از زندان داشته باشد که یکتا نیستند اما تعداد محدودی در بازی وجود دارد. استاندارد ERC1155 همه‌ی این نیازها را برطرف می‌کند. همه‌ی این چند نوع توکن می‌توانند همزمان در یک قرارداد هوشمند وجود داشته باشند.

در این استاندارد متدهایی برای تعریف نوعی توکن با تعداد مشخص وجود دارد. اگر نیاز به توکنی تعویض ناپذیر باشد تعداد آن یک قرارداده می‌شود. همچنین متدهایی برای ارسال تعداد مشخص از چند نوع توکن مختلف در یک تراکنش، دادن وکالت توکن‌ها به آدرس دیگر و گرفتن موجودی یک آدرس در این استاندارد وجود دارد.

اطلاعات دقیق‌تر در مورد این استاندارد را نیز می‌توان در وبسایت اتریوم [۹] یا اپن‌زپلین [۱۰] مشاهده کرد.

[9] https://ethereum.org/en/developers/docs/standards/tokens/erc-1155
[10] https://docs.openzeppelin.com/contracts/4.x/api/token/erc1155

فصل ۳

آشنایی با ابزارهای توسعه

در تمام ابزارهای ذکر شده در ادامه این متن حتما باید به ورژن هر کدام دقت شود، ورژن‌ها باید با یکدیگر همخوانی داشته باشند در غیر این صورت مشکلاتی در کامپایل و اجرای برنامه به وجود می‌آید که به راحتی قابل رفع کردن نیستند. در انجام این پروژه عدم همخوانی ورژن‌های مختلف ابزارها با یکدیگر باعث ایجاد مشکلات فراوانی شد، به همین دلیل ورژن مورد نیاز هر ابزار در توضیحات پروژه ذکر شده است.

۱.۳ ابزارهای ساده

- **ویرایشگر**

برای برنامه نویسی این قرارداد هوشمند از ویرایشگر VSCode با نصب پلاگین مربوط به Solidity [۱] استفاده شده است. این پلاگین با یافتن اشتباه‌ها پیش از کامپایل و راهنمایی در نوشتن کد قرارداد کمک شایانی به افزایش سرعت توسعه می‌کند.

- **ورژن‌کنترل**

این پروژه از روز نخست به صورت متن‌باز توسعه یافته، برای توسعه یک پروژه به صورت متن‌باز اولین ابزار مورد نیاز یک برنامه ورژن کنترل است که نسخه‌های متفاوت و تغییر یافته کدها را به صورت مرتب

[1] https://marketplace.visualstudio.com/items?itemName=JuanBlanco.solidity

نگهداری کند. برای این منظور از گیت‌هاب استفاده شده.

- **پکیج‌های Node و NPM**

از آنجایی که کدهای سالیدیتی در واقع جاوااسکریپت هستند، به ابزارهای توسعه اپلیکیشن‌های جاوااسکریپت برای توسعه سالیدیتی نیاز است. ابزارهایی مانند Node برای کامپایل کردن برنامه‌های جاوااسکریپت و npm که مدیریت پکیج‌های جاوااسکریپتی که نصب می‌شود را به عهده دارد.

۲.۳ کیف پول متامسک

کیف پول دیجیتال متامسک از پرکاربردترین کیف پول‌ها برای ارتباط برقرار کردن با اپلیکیشن‌های غیرمتمرکز و Web3 است. کاپو نیز برای امضای تراکنش‌ها و ایجاد ارتباط با شبکه بلاکچین از کیف پول متامسک استفاده می‌کند. برای انجام صحیح این عملیات کاربر باید از پیش کیف پول متامسک را نصب کرده باشد و سپس با انتخاب گزینه Connect Wallet، کاپو درخواست اتصال به کیف پول و دریافت آدرس کاربر را به متامسک ارسال میکند، متامسک نیز پس از دریافت درخواست کاپو از کاربر اجازه اتصال به اپلیکیشن را میگیرد و در صورت تایید کاربر آدرس کیف پول را به کاپو می‌دهد.

از این پس هرگاه که کابر بخواهد در کاپو تراکنشی از جمله ساخت توکن جدید یا انتقال یک توکن به آدرس دیگر را انجام دهد کاپو از متامسک درخواست می‌کند که با کلید خصوصی[۲] کاربر آن تراکنش را امضا کند، متامسک از کاربر تایید تراکنش را میگیرد و امضا را انجام می‌دهد و تراکنش به شبکه بلاکچین ارسال می‌شود.

۳.۳ فریمورک‌ها و کتابخانه‌ها

به دلیل تازگی بحث توسعه اپلیکیشن‌های غیرمتمرکز ابزارهای کمی در این زمینه وجود دارند و همین ابزارها هم معمولا مشکلاتی دارند و به بلوغ کامل نرسیده‌اند. اما با توجه به این که اکثر ابزارها و فریمورک‌ها و کتابخانه‌های توسعه اپلیکیشن‌های غیرمتمرکز متن‌باز هستند، سرعت رشد و تکامل بالایی دارند و به کمک توسعه‌دهندگان[۳] این

[2] Private key
[3] Developers

حوزه، هر روز نسبت به روز گذشته پیشرفت می‌کنند.

برای توسعه این پروژه از فریمورک Truffle [۴]، کتابخانه‌ی OpenZeppelin [۵]، کتابخانه‌ی Web3JS [۶] استفاده شده است. در این قسمت به توضیح هر یک از این موارد پرداخته می‌شود.

۱.۳.۳ فریمورک Truffle

این فریمورک ابزارهای اولیه برای ساخت، کامپایل، تست، دیپلوی و مایگریشن قراردادهای هوشمند به زبان سالیدیتی را فراهم می‌کند. پس از نصب این ابزار با اجرای دستور truffle init می‌توان یک پروژه جدید ترافل ساخت، همچنین میتوان با استفاده از دستور truffle unbox از یکی از تمپلیت‌های آماده ترافل استفاده کرد.

```
→  new-project truffle init

Starting init...
================

> Copying project files to /Users/AminBSHR/Desktop/Thesis/new-project

Init successful, sweet!

Try our scaffold commands to get started:
  $ truffle create contract YourContractName # scaffold a contract
  $ truffle create test YourTestName         # scaffold a test

http://trufflesuite.com/docs
```

شکل ۱.۳: اجرای دستور truffle init

پس از ساخت پروژه با اجرای دستور truffle develop و یا truffle console میتوان وارد خط فرمان ترافل شد.

دستورات لازم برای اجرای تست‌ها، کامپایل کردن قراردادهوشمند یا دیپلوی آن روی شبکه مورد نظر از طریق این خط فرمان قابل اجرا هستند. این پلتفرم ابزارهای فراوانی را در اختیار توسعه دهنده قرار می‌دهد که با تعداد بیشتری از آن‌ها در بخش پیاده‌سازی و دیپلوی کاپو آشنا می‌شویم. همچنین از بزرگترین مزایای استفاده از این فریمورک برقراری ارتباط بسیار آسان با ابزارهای دیگر مانند Ganache و Drizzle است.

[4] https://trufflesuite.com
[5] https://openzeppelin.com/contracts
[6] https://github.com/ChainSafe/web3.js

شکل ۲.۳: اجرای دستور truffle develop

۲.۳.۳ کتابخانه OpenZeppelin

یکی از معروف‌ترین کتابخانه‌های قراردادهای هوشمند و استانداردهایشان است. قراردادها و استانداردهای موجود در این کتابخانه کاملا تست شده، داکیومنت شده، ایمن و پایه بسیاری از قراردادهای هوشمند بر بستر بلاکچین هستند. استانداردهای ذکر شده در این متن مانند، ERC20، ERC721، ERC1155 به همراه تعداد زیادی استانداردهای دیگر در این کتابخانه پیاده‌سازی شده‌اند.

در کاپو نیز از استاندارد ERC721 پیاده‌سازی شده در این کتابخانه استفاده شده است. برای استفاده از قراردادهای اپن‌زپلین در قدم اول باید این کتابخانه به کمک دستور npm install @openzeppelin/contracts نصب شود. پس از نصب کتابخانه، می‌توان از قراردادهای آن ارث‌بری کرد، در قطعه کد زیر مشاهده می‌شود که کاپو چگونه از قرارداد ERC721 موجود در اپن‌زپلین و همچنین یک قراردادهوشمند به اسم Helper که در همین پروژه نوشته شده ارث‌بری کرده است.

```
2  pragma solidity >=0.4.22 <0.9.0;
3  import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
4  import "./Helper.sol";
5
6  contract Cappu is ERC721, Helper {
7      constructor() ERC721("Cappu", "CAPU") {}
8  }
```

شکل ۳.۳: ارث‌بری از استاندارد ERC721 پیاده‌سازی شده توسط OpenZeppelin

۳.۳.۳ کتابخانه Web3JS

تراکنش‌های با یک قرارداد هوشمند می‌تواند به ۲ حالت باشد. در حالت اول فقط اطلاعات شبکه بلاکچین خوانده می‌شود و حالت[۷] آن تغییری داده نمی‌شود، متدهای از این جنس از نوع view یا pure هستند. حالت دوم تراکنش‌هایی هستند که باعث تغییر اطلاعات شبکه بلاکچین می‌شوند.

فرانت‌اند یک اپلیکیشن غیرمتمرکز برای انجام نوع اول تراکنش‌های نهایتا فقط به آدرس کاربر نیاز دارد که اطلاعات مربوط به او را از قرار داد بگیرد. در حالت دوم نیاز است که تراکنشی بر روی شبکه ثبت شود که نیازمند امضا شدن تراکنش توسط کلید خصوصی کاربر، پرداخت کارمزد تراکنش و ارسال آن به نودهای شبکه است.

کتابخانه‌ی Web3JS به توسعه دهنده کمک می‌کند که فرانت‌اند اپلیکیشن را به کیف پول دیجیتال کاربر و شبکه بلاکچین متصل کند. با ایجاد این اتصال آدرس کابر توسط کیف پول دیجیتال در اختیار فرانت‌اند قرار می‌گیرد و هرگاه که فرانت‌اند بخواهد تراکنشی را روی شبکه ارسال کند نیز از کیف پول کاربر می‌خواهد که با داشتن کلید خصوصی کاربر آن تراکنش را امضا و روی شبکه ارسال کند. طبیعتا کیف پول کاربر برای انجام هر یک از این مراحل از کاربر درخواست تاییدیه می کند.

۴.۳ شبکه لوکال برای توسعه

برای توسعه یک اسمارت کانترکت نیاز است که پس از هر تغییر کامپایل و روی یک شبکه بلاکچین دیپلوی شود، به نحوی که فرانت‌اند اپلیکیشن و همچنین کیف پول متامسک بتوانند به آن متصل شوند. از شبکه اصلی نمی‌توان استفاده کرد زیرا هر دیپلوی روی شبکه اصلی هزینه‌ای خواهد داشت و دیپلوی‌های پیاپی روی شبکه

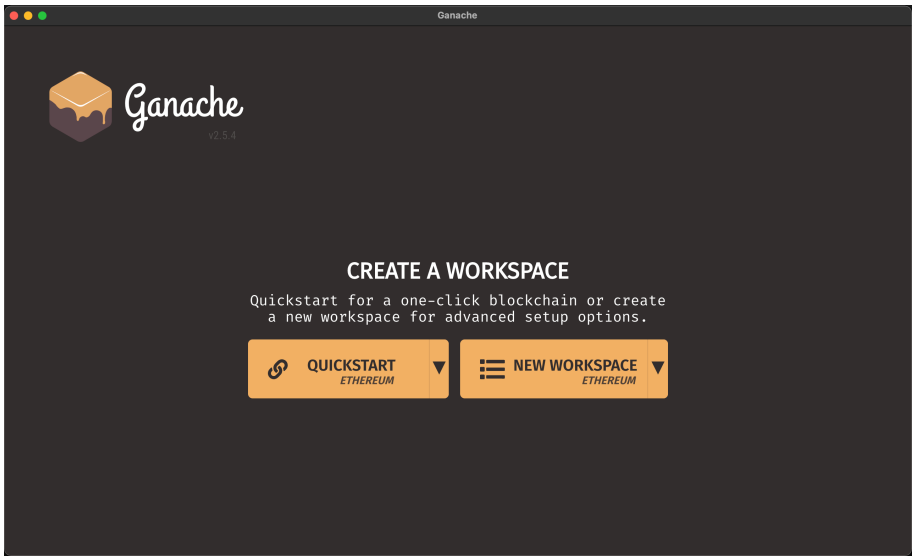
[7] State

امکان پذیر نخواهد بود. اگر بخواهیم برای توسعه از شبکه تستی هم استفاده کنیم گرچه هزینه‌ای نخواهد داشت اما بسیار زمان‌بر خواهد بود، گرچه انجام تراکنش‌ها روی شبکه تستی معمولا سریعتر از شبکه اصلی انجام می‌شود اما همچنان توسعه دهنده زمان زیادی را برای هر دیپلوی صرف خواهد کرد.

راه حل این مشکل این است که توسعه دهنده روی ماشین خودش یک شبکه لوکال داشته باشد که بتواند بلافاصله پس از ایجاد یک تغییر روی قراردادهوشمند آن را کامپایل و دیپلوی کند. ترافل باید بتواند به این شبکه لوکال متصل شود و قرارداد را روی آن دیپلوی کند. فرانت‌اند ومتامسک نیز باید بتوانند به این شبکه متصل شوندکه با قراردادهوشمند ارتباط برقرار کنند.

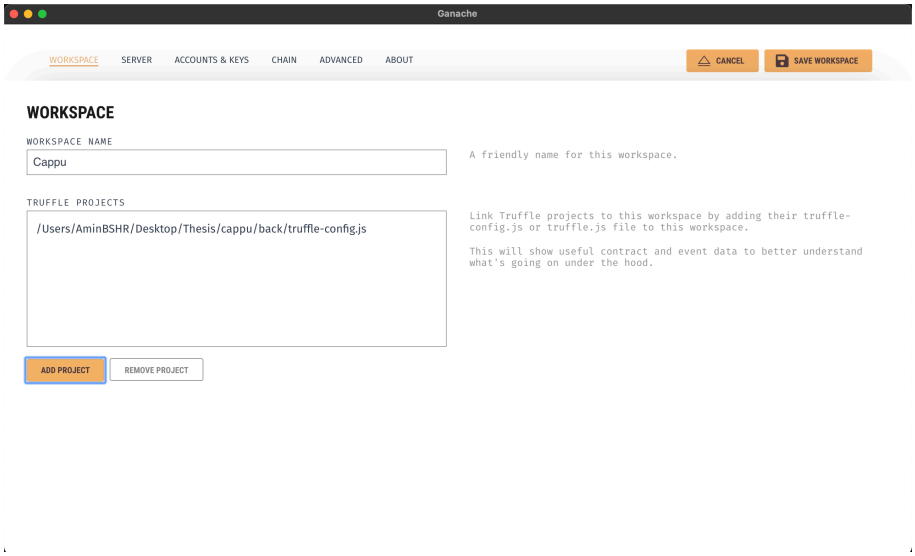
اگرچه ابزارهای زیادی برای ساخت این شبکه لوکال وجود دارند، اما یکی از بهترین و راحت‌ترین ابزارها برای این منظور برنامه‌ی Ganache هست. این ابزار با توجه به این که متعلق به اکوسیستم Truffle هست به آسانی به آن متصل می‌شود و با اضافه کردن آدرس آن به شبکه‌های متامسک، این کیف پول هم به شبکه لوکال متصل می‌شود. جزئیات ساخت شبکه لوکال و اتصال ترافل و متامسک به آن به ترتیب زیر است.

پس از نصب برنامه Ganache باید یک محیط توسعه Ethereum ساخته شود. برای انجام این کار گزینه New workspace (Ethereum) انتخاب می‌شود.



شکل ۴.۳: صفحه اول Ganache

سپس در صفحه باز شده نام محیط توسعه وارد، فایل truffle-config.js مربوط به پروژه مورد نظر انتخاب و دکمه save workspace زده می‌شود.



شکل ۵.۳: ساخت شبکه جدید در گاناچه

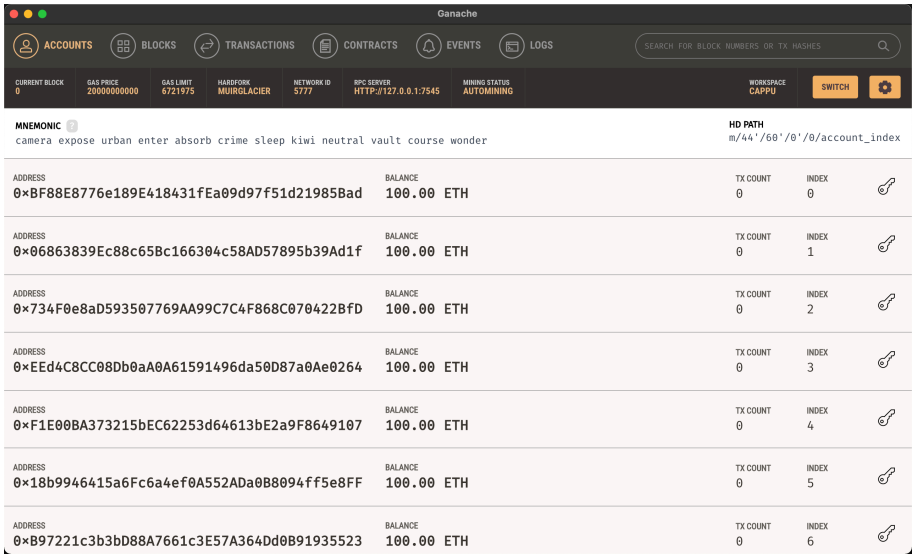
پس از انجام این مراحل محیط توسعه ساخته شده است و می‌توان جزئیات شبکه لوکال را مشاهده کرد.

از آنجایی که در مرحله قبل برای ساخت این محیط توسعه فایل truffle-config.js پروژه انتخاب شد، حال اگر دستورات truffle console یا هر دستور دیگری مانند migrate بدون انتخاب شبکه بلاکچین خاصی اجرا شود به صورت پیش‌فرض روی این شبکه لوکال انجام می‌شود.

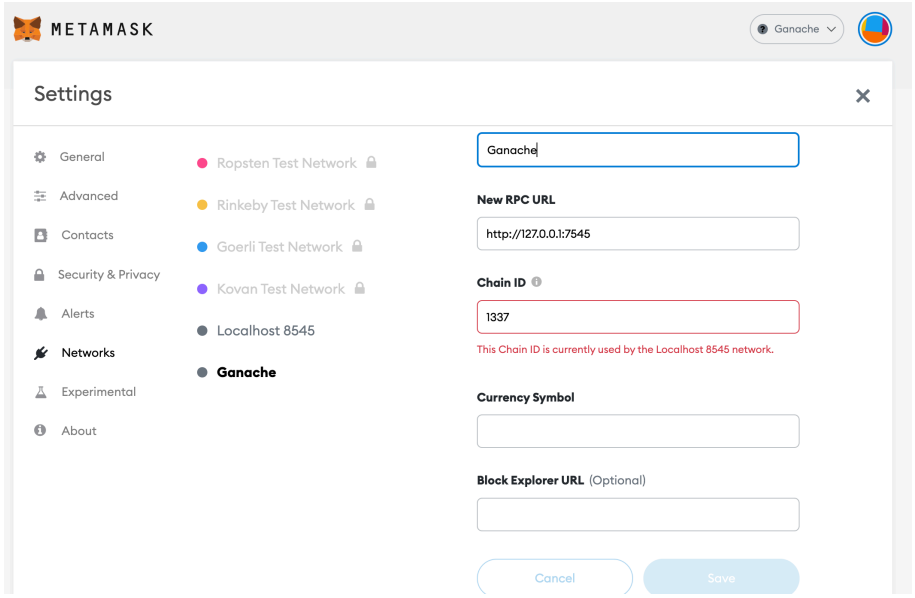
حال فقط باید متامسک نیز به این شبکه لوکال متصل شود. برای انجام این کار پس از نصب افزونه‌ی متامسک روی مرورگر کروم، در قسمت تنظیمات[8] و سپس شبکه‌ها[9] یک شبکه جدید با جزئیات زیر ساخته می شود، همانطور که در تصویر بالا ۶.۳ مشخص است اطلاعات شبکه لوکال در صفحه اصلی Ganache قابل مشاهده هستند.

پس از ذخیره شبکه جدید کافیست که برای توسعه شبکه Ganache انتخاب شود. همچنین باید یکی از آدرس‌هایی که در صفحه اصلی Ganache نمایش داده می‌شوند به عنوان کیف پول در متامسک وارد شود. برای انجام این کار علامت کلید کنار یکی از آدرس‌های نمایش داده شده در صفحه اصلی Ganache انتخاب می‌شود و به کمک کلید اختصاصی نمایش داده شده کیف پول در متامسک وارد می شود.

[8]Settings
[9]Networks



شکل ۶.۳: مشاهده جزئیات شبکه ساخته شده



شکل ۷.۳: تنظیمات شبکه‌های متامسک

فصل ۴

پیاده‌سازی

۱.۴ نوشتن کد قرارداد

در این بخش به بررسی مراحل و نحوه نوشتن کد قراردادهوشمند پرداخته می‌شود.

۱.۱.۴ نیازمندی‌های قراردادهوشمند

نیازمندی‌های اصلی کاپو به ترتیب زیر است.

- هر آدرس در شبکه بتواند یک داده‌ی متنی را به آسان‌ترین و کم هزینه‌ترین روش ممکن به یک توکن NFT تبدیل کند.
- هر آدرس بتواند توکن‌های خود را به اشخاص دیگر انتقال دهد یا در بازارهای معاملات NFT بفروشد.
- در صفحه اول وبسایت تعداد کل توکن‌های ساخته شده تا به حال و تعداد کل دارندگان توکن نمایش داده شود.
- قابلیت‌های قراردادهوشمند تست شده باشد.

۲.۱.۴ ارث‌بری

با توجه به مزایای ذکر شده در مورد استانداردسازی قراردادهای هوشمند، انتخاب درستی است که برای پیاده‌سازی این کاربری از یکی از استانداردها استفاده شود. ارثبری از استانداردهای یک کتابخانه متن‌باز مزایای زیر را فراهم می‌کند.

- به دلیل وجود کدهای پایه به صورت آماده سرعت توسعه پروژه افزایش می‌یابد.
- ارتباط دیگر پروژه‌ها با پروژه کاپو به راحتی انجام می‌شود.
- امینت قرارداد و درستی آن حداقل در سطوح پایه‌ای تا حد خوبی تضمین شده است.

قراردادهوشمند کاپو از استاندارد ERC۷۲۱ پیاده‌سازی شده در کتابخانه اپن‌زپلین [1] ارث‌بری میکند که یکی از معروف ترین کتابخانه‌های پیاده کننده استانداردهای قرارداد هوشمند است.

۳.۱.۴ توجه به هزینه تراکنش و نوع توابع

در نوشتن یک قراردادهوشمند باید به نکات زیر توجه کنیم.

- میزان حافظه‌ای که اشغال می‌کنیم.
- حجم بایت‌کد.
- میزان عملیات هر متد، به خصوص متدهایی که مکررا مورد استفاده کاربر قرار می‌گیرند.
- نوع هر متد، که مشخص می‌کند هر متد تا چه حد روی شبکه بلاکچین تغییر ایجاد می‌کند.

توجه نکردن به هریک از این موضوعات باعث می‌شود که قراردادهوشمند به اندازه کافی بهینه عمل نکند و کاربر وادار به پرداخت gasfee یا هزینه تراکنش بیشتر شود. یکی از مهمترین نکاتی که برای بهینه‌تر رفتار کردن قراردادهوشمند باید به آن توجه کنیم نوع هر متد است.

[1] https://github.com/OpenZeppelin/openzeppelin-contracts

اگر متدی از نوع pure تعریف شود به این معنی است که به هیچ اطلاعاتی از شبکه بلاک‌چین نیاز ندارد و همه‌ی اطلاعاتی که لازم دارد را در اسکوپ [۲] خودش دارد. اگر متدی از نوع view باشد به این معنی است که به اطلاعاتش روی شبکه بلاکچین نیاز دارد اما فقط می‌خواهد که آن‌ها را بخواند و نمیخواهد تغییری در آن‌ها ایجاد کند. این دو نوع متد نیازی به پرداخت کارمزد تراکنش توسط کاربر ندارند، اما اگر در تعریف متدی ذکر نشود که یکی از این دو نوع است، اینطور در نظر گرفته می‌شود که نیاز به بروزرسانی اطلاعاتش در شبکه بلاکچین دارد و از کاربری که آن را فراخوانی کرده است هزینه تراکنش دریافت می‌شود.

۴.۱.۴ جزئیات فنی پیاده‌سازی

مینت کردن در این قرارداد به آدرس‌های مشخص محدود نیست و همه می‌توانند توکن بسازند. بسیاری از قراردادها برای صرفه‌جویی در هزینه تراکنش کاربران اکثر اطلاعات مربوط به توکن‌ها را در قرارداد نگه نمیدارند و فقط داده‌های بسیار مهم توکن را در شبکه بلاکچین نگهداری می کنند. از آنجایی که کاپو یک قرارداد همه منظوره است و ممکن است استفاده‌های فراوانی داشته باشد، تصمیم‌گیری این مورد به عهده کاربر قرارداد گذاشته می‌شود.

در کاپو آیدی هر توکن از hash داده‌های توکن به دست می‌آید. این نحوه عملکرد چند مذیت ایجاد می‌کند. به این ترتیب هیچ دو توکنی نمی‌توانند داده‌های یکسان داشته باشند، زیرا در این صورت آیدی آن‌ها باید یکسان باشد و این امکان پذیر نیست زیرا آیدی توکن‌ها یکتاست. همچنین آیدی توکن‌ها دیگر ترتیبی نخواهند بود و ترتیب ساخت توکن‌ها مشخص نخواهد بود.

در یک قرارداد ERC۷۲۱ استاندارد فقط آیدی توکن‌ها ذخیره می‌شود. در کاپو علاوه بر آیدی توکن‌ها یک map از آیدی توکن‌ها به داده‌ی آن‌ها با نام tokenDatas نیز نگهداری می‌شود. همچنین در کاپو map دیگری نیز از آدرس به لیست توکن‌های آن آدرس با نام ownerTokens نگهداری می‌شود. متغیر اول کمک می‌کند که با داشتن آیدی یک توکن به راحتی داده‌های آن توکن به دست آورده شوند. متغیر دوم نیز کمک می‌کند که به راحتی بتوان توکن‌های یک آدرس را به دست آورد. دومتغیر دیگر با نام‌های numberOfTokenHolders و numberOfMintedTokens نیز در کاپو نگه‌داشته می‌شوند که برای نمایش آمار استفاده از قرارداد در صفحه اصلی اپلیکیشن مورد استفاده قرار می‌گیرند.

[2] Scope

متد mint به نحوی نوشته شده است که برای عموم قابل استفاده باشد. پس از محاسبه hash داده‌ی توکن از آن به عنوان آیدی توکن استفاده میکند، توکن را می‌سازد و متغیرهای tokenDatas و numberOfMintedTokens را بروزرسانی می‌کند.

```
14      function mint(string memory data) public {
15          uint256 theHash = uint256(keccak256(abi.encode(data)));
16          _safeMint(msg.sender, theHash);
17          _tokenDatas[theHash] = data;
18          _numberOfMintedTokens++;
19      }
```

شکل ۱.۴: پیاده‌سازی تابع mint

متد afterTokenTransfer از استاندارد ERC۷۲۱ به نحوی بازنویسی [۳] شده است که پس از هر انتقال توکن با بررسی آدرس‌های مبدا و مقصد، متغیرهای numberOfTokenHolders و numberOfMinted-Tokens و ownerTokens را بروزرسانی کند.

```
21      function _afterTokenTransfer(
22          address from,
23          address to,
24          uint256 tokenId
25      ) internal virtual override {
26          if (from != address(0)) {
27              _ownerTokens[from] = removeItemFromArray(
28                  tokenId,
29                  _ownerTokens[from]
30              );
31              if (_ownerTokens[from].length == 0) {
32                  _numberOfTokenHolders--;
33              }
34          }
35          if (to != address(0)) {
36              _ownerTokens[to].push(tokenId);
37              if (_ownerTokens[to].length == 1) {
38                  _numberOfTokenHolders++;
39              }
40          }
41      }
```

شکل ۲.۴: پیاده‌سازی تابع afterTokenTransfer

[3] Overwrite

متد جدیدی با نام getUserTokens نیز نوشته شده است که در استاندارد ERC۷۲۱ به صورت پیش‌فرض وجود ندارد. این متد با گرفتن یک آدرس و استفاده از ownerTokens و tokenDatas دو خروجی برمی‌گرداند، لیستی از آیدی توکن‌های آدرس و لیستی از داده‌های توکن‌های آدرس.

```
43      function getUserTokens(address user)
44          public
45          view
46          returns (uint256[] memory, string[] memory)
47      {
48          uint256[] memory tokens = _ownerTokens[user];
49          string[] memory datas = new string[](tokens.length);
50          for (uint256 i = 0; i < tokens.length; i++) {
51              datas[i] = _tokenDatas[tokens[i]];
52          }
53          return (tokens, datas);
54      }
```

شکل ۳.۴: پیاده‌سازی تابع getUserTokens

همچنین از آنجایی که سالیدیتی به طور پیش‌فرض امکان حذف یک داده از یک آرایه با داشتن مقدار آن را ندارد، عدم وجود این قابلیت هزینه‌بر بودن آن است، در سالیدیتی توسعه دهندگان به استفاده از map و دوری از array ها تشویق می‌شوند. اما برای نمایش نحوه ارث‌بری از دو یا چند قرارداد پدر، برای کاپو یک قرارداد به نام Helper نوشته شد که این قابلیت را فراهم می‌کند. کاپو علاوه بر ERC721 از قرارداد Helper نیز ارث‌بری می‌کند.

۲.۴ نوشتن و اجرای تست‌ها

پیش‌تر اشاره شد که از مزیت‌های ارث‌بری از کتابخانه‌های متن‌باز معروف این است که احتمال وجود خطا و مشکل امنیتی به شدت کمتر می‌شود. یکی از دلایل این مسئله این است که این کتابخانه‌ها پوشش تستی به شدت بالایی دارند. به همین دلیل می‌توان تا حدی به عملکرد قرارداد پدر اطمینان خاطر داشت و بیشتر روی تست کردن قابلیت‌های اضافه شده در قراردادهوشمند فرزند تمرکز داشت.

در کاپو برای هر عملکرد قرارداد تست نوشته شده است. یکی از ساده‌ترین تست‌های نوشته شده تست فرآیند ساخت یک توکن است که در آن پس از دیپلوی قرارداد با فراخوانی متد mint یک توکن ساخته می‌شود و سپس

```
contract Helper {
    function removeItemFromArray(
        uint256 valueToFindAndRemove,
        uint256[] memory array
    ) internal pure returns (uint256[] memory) {
        uint256[] memory auxArray = new uint256[](array.length - 1);
        uint8 found = 0;
        for (uint256 i = 0; i < array.length; i++) {
            if (array[i] != valueToFindAndRemove) {
                auxArray[i - found] = array[i];
            } else {
                found = 1;
            }
        }
        if (found == 0) {
            return array;
        }
        return auxArray;
    }
}
```

شکل ۴.۴: پیاده‌سازی قرارداد Helper

با فراخوانی متد balanceOf دارایی آدرس سازنده توکن بررسی می‌شود و انتظار می‌رود که پس از ساخت یک توکن، دارایی آدرس سازنده توکن یک باشد. این تست را می‌توان در تصویر زیر مشاهده کرد.

```
contract("Cappu", (accounts) => {
  it("should mint a token", async () => {
    const cappu = await Cappu.deployed();

    await cappu.mint("Hey there!", { from: accounts[0] });

    const balance = await cappu.balanceOf(accounts[0], {
      from: accounts[0],
    });

    assert.equal(balance, 1);
  });
});
```

شکل ۵.۴: نمونه یکی از تست‌های قرارداد کاپو

پس از نوشته شدن تست‌ها می‌توان آن‌ها را با اجرای دستور truffle test اجرا کرد. این دستور پس از اجرای تست‌ها نتیجه و زمان اجرای هر تست را به عنوان خروجی نمایش می‌دهد. نمونه اجرای این دستور را می‌توان در تصویر زیر مشاهده کرد.

```
➜  back git:(main) ✗ truffle test
Using network 'test'.


Compiling your contracts...
===========================
> Compiling ./contracts/Cappu.sol
> Artifacts written to /var/folders/rp/gbmd1rwd0p1325ck73ckz_rc0000gn/T/test--15817-mpJlgp9Ttlws
> Compiled successfully using:
   - solc: 0.8.10+commit.fc410830.Emscripten.clang


  Contract: Cappu
    ✓ should return correct homepage info (6505ms)

  Contract: Cappu
    ✓ should mint a token (1980ms)

  Contract: Cappu
    ✓ should transfer a token (2301ms)


  3 passing (13s)
```

شکل ۶.۴: نمونه خروجی اجرای تست‌های قرارداد

۳.۴ دیپلوی قرارداد روی شبکه تستی Ropsten

تا اینجا قراردادهوشمند نوشته و تست شده است، در این مرحله روی شبکه تستی Ropsten دیپلوی می‌شود. فرآیند دیپلوی شدن کاپو به کمک فریمورک ترافل قدم به قدم شرح داده می‌شود.

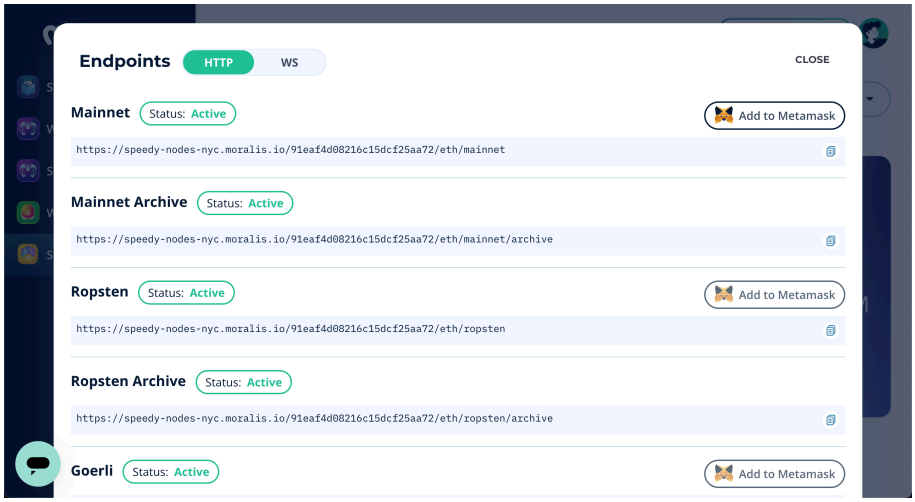
۱.۳.۴ یافتن آدرس یکی از نودهای شبکه برای ارسال تراکنش دیپلوی قرارداد به آن

آدرس نودهای یک شبکه بلاکچین همه به صورت عمومی در دسترس هستند زیرا نودها باید بتوانند یکدیگر را ببینند. راه‌های زیادی برای به دست آوردن آدرس یک نود وجود دارد. یکی از آسان‌ترین راه‌های به دست آوردن آدرس یکی از نودهای شبکه مراجعه به وبسایت ماینر است. برای این پروژه از وبسایت Moralis [۴] برای پیدا کردن آدرس نود شبکه استفاده شد.

۲.۳.۴ اضافه شدن اطلاعات شبکه مورد نظر به تنظیمات ترافل

هنگامی که به کمک دستور truffle init یک پروژه ترافل ساخته می‌شود، فایلی با نام truffle-config.js ساخته می‌شود. تنظیمات مربوط به ترافل در این فایل نوشته شده‌است. برای این که ترافل شبکه مورد نظر را

[4] https://moralis.io



شکل ۷.۴: دریافت آدرس یکی از نودهای شبکه از وبسایت Moralis

بشناسد باید اطلاعات آن شبکه در این فایل نوشته و شبکه‌ی جدیدی تعریف شود. برای تعریف شبکه از آدرسی که در گام قبل به دست آمد استفاده می‌شود و مانند تصویر زیر شبکه‌ی جدیدی تعریف می‌شود.

```
38  networks: {
39    ropsten: {
40      provider: () =>
41        new HDWalletProvider(
42          mnemonic,
43          `https://speedy-nodes-nyc.moralis.io/91eaf4d08216c15dcf25aa72/eth/ropsten`
44        ),
45      network_id: 3, // Ropsten's id
46      gas: 8000000, // Ropsten has a lower block limit than mainnet
47      gasPrice: 100000000000,
48      confirmations: 2, // # of confs to wait between deployments. (default: 0)
49      timeoutBlocks: 200, // # of blocks before a deployment times out  (minimum/default: 50)
50      skipDryRun: true, // Skip dry run before migrations? (default: false for public nets )
51    }
52  },
```

شکل ۸.۴: اضافه کردن شبکه Ropsten به شبکه‌های ترافل

۳.۳.۴ آماده شدن mnemonics

برای انجام این پروژه به کمک دستور npm mnemonics یک آدرس تستی ساخته می‌شود. این دستور، mnemonics متناسب با این آدرس را به عنوان خروجی می‌دهد. دقت کنید که برای دیپلوی روی شبکه اصلی[۵] حتما باید از mnemonics مربوط به یک کیف پول واقعی استفاده شود و اطلاعات ان در اختیار کسی قرار نگیرد.

[5] Mainnets

```
→  cappu git:(main) ✗ npx mnemonics
flavor bleak joy tired bid habit regret prison nasty acoustic amount thought
```

شکل ۹.۴: ایجاد mnemonics تستی

۴.۳.۴ استفاده از کیف پول ایجاد شده در تنظیمات ترافل

ترافل برای این که بتواند از کیف‌پول برای انجام تراکنش‌ها استفاده کند باید به mnemonics یا کلید خصوصی آن دسترسی داشته باشد. به این منظور فایلی با نام secrets.json در دایرکتوری اصلی برنامه ساخته می‌شود و mnemonics کیف پول به شکل زیر در آن قرار داده می‌شود.

```
1  {
2    "mnemonic": "sun seat live wealth pistol bubble ...",
3  }
4
```

شکل ۱۰.۴: قراردادن mnemonics در فایل secrets.json

سپس در تنظیمات ترافل باید ذکر شود که می‌تواند آدرس کیف‌پول را در این آدرس پیدا کند.

```
25
26  const mnemonic = require("./secrets.json").mnemonic;
27
```

شکل ۱۱.۴: معرفی فایل secrets.json در تنظیمات ترافل

۵.۳.۴ نصب کیف‌پول hdwallet

ترافل برای استفاده از mnemonics کیف پول ما نیاز به نصب پکیج hdwallet-provider دارد، این پکیج کاربری‌های یک کیف پول دیجیتال از جمله امضا و ارسال تراکنش بر روی شبکه بلاکچین را در اختیار ترافل قرار می‌دهد. این پکیج با اجرای دستور npm install –save-dev @truffle/hdwallet-provider نصب می‌شود. پس از نصب کیف پول در تنظیمات ترافل در فایل truffle-config.js ذکر می‌شود که از این کیف‌پول استفاده شود.

```
20
21  const HDWalletProvider = require("@truffle/hdwallet-provider");
22
```

شکل ۱۲.۴: استفاده از کیف‌پول hdwallet در تنظیمات ترافل

۶.۳.۴ انتخاب شبکه اضافه شده

حال هنگام ورود به خط فرمان ترافل مانند تصویر زیر شبکه مورد نظر انتخاب می‌شود.

```
➜  back git:(main) ✗ truffle console --network ropsten
truffle(ropsten)>
```

شکل ۱۳.۴: ورود به خط فرمان ترافل با انتخاب شبکه Ropsten

۷.۳.۴ بررسی آدرس کیف‌پول و موجودی آن

برای دیپلوی یک قرارداد هوشمند باید آدرس دیپلوی کننده آن بتواند هزینه تراکنش دیپلوی را پرداخت کند. در صورتی که دیپلوی بر روی یک شبکه تستی انجام می‌شود باید با استفاده از یک faucet روی شبکه تستی به میزان کافی پول تستی دریافت شود.

برای دریافت آدرس‌های کیف‌پول از دستور زیر در خط فرمان ترافل استفاده می شود.

```
truffle(ropsten)> await web3.eth.getAccounts()
[
  '0xF51f5f41BfA8ADa57a43862cBc18dA4750AecB4c',
  '0x909ebC92395FC4335c35894C7DDc8bfFFDCeEF06',
  '0x48156708DF687C7a8F97C951b5E734E132e891D1',
  '0xF1C6c91D80032528e2C01F73DAd588D11DA0f17d',
  '0xA6f899d10B4E1c1195AFD1C6f29E4e539C828450',
  '0xB63191Dd13637c024C7F1F339F254F0F13d4bB34',
  '0x1699Ba468F7E5af64f510B323537bbcd107373F9',
  '0x6eDd855A6D2d3De5D96749e1bD3E9580c33468E7',
  '0x8A97C0bfC3086DFcd9E1B25D69A1A238A1290BE6',
  '0xc4838dF4d46862d1226BDC409EbE8395cA6fE703'
]
```

شکل ۱۴.۴: دریافت آدرس‌های کیف‌پول در خط فرمان ترافل

برای دریافت مانده حساب آدرس، دستور زیر در خط فرمان ترافل اجرا می‌شود.

```
truffle(ropsten)> await web3.eth.getBalance("0xF51f5f41BfA8ADa57a43862cBc18dA4750AecB4c")
'790887817599784390'
truffle(ropsten)>
```

شکل ۱۵.۴: دریافت موجودی کیف‌پول در خط فرمان ترافل

۸.۳.۴ دیپلوی قراردادهوشمند روی شبکه بلاکچین

پس از اطمینان از توانایی پرداخت کارمزد تراکنش با استفاده از دستور migrate در خط فرمان ترافل قراردادهوشمند روی شبکه بلاکچین دیپلوی می‌شود.

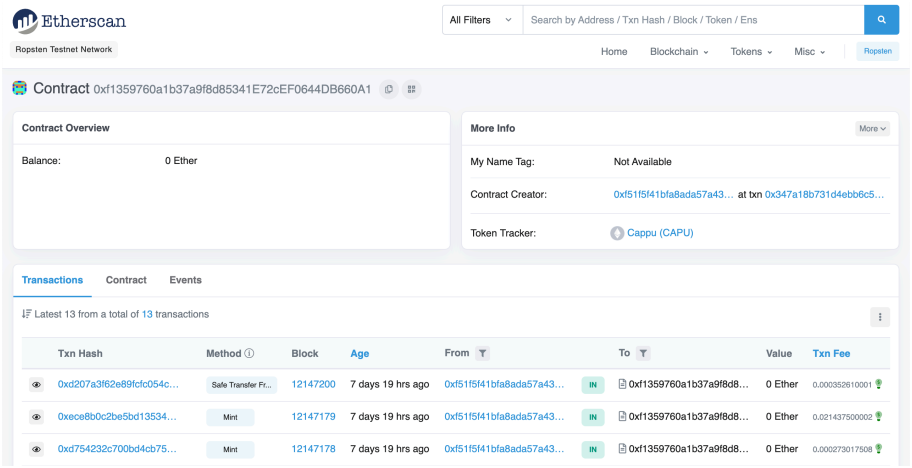
۹.۳.۴ اطمینان از صحت دیپلوی قراردادهوشمند

س از اتمام دیپلوی قرارداد هوشمند برای اطمینان از به درستی انجام شدن فرآیند دیپلوی قرارداد، می‌توان از جستجوگرهای شبکه[۶] بلاکچین استفاده کرد. برای مثال قرارداد هوشمند کاپو بر روی شبکه Ropsten دیپلوی شده است، که با رفتن به وبسایت اتراسکن [۷] و قراردادن آن روی شبکه Ropsten می‌توان قرارداد دیپلوی شده و تراکنش‌های آن را مشاهده کرد.

۴.۴ توسعه فرانت، اتصال به قراردادهوشمند و فرآیند دیپلوی

برای توسعه فرانت‌اند اپلیکیشن، React به عنوان فریمورک مورد استفاده انتخاب شد. ترکیب این فریمورک با استفاده از کتابخانه material-ui که کمک می‌کند در زمان کوتاه بتوان ظاهری زیبا و یکدست در اپلیکیشن ایجاد کرد و کتابخانه Web3JS که فرانت‌اند را به کیف پول کاربر و شبکه بلاکچین متصل می‌کند، همه‌ی قابلیت‌های مورد نیاز برای توسعه یک فرانت‌اند زیبا و کارآمد را در اختیار توسعه دهنده قرار می‌دهد.

[6] Block Explorers
[7] `https://etherscan.io`



شکل ۱۶.۴: مشاهده قراداد کاپو در Etherscan روی شبکه Ropsten

در پوشه اصلی فرانت‌اند فایلی با عنوان config.js وجود دارد. در این فایل علاوه بر ABI قراردادهوشمند سایر اطلاعات مورد نیاز مانند آدرس شبکه، آدرس قرارداد در شبکه و نام شبکه مورد نظر نیز ذخیره می‌شود. هنگام توسعه باید دقت شود که این فایل به قرارداد روی شبکه لوکال متصل شود.

برای استفاده از Web3JS و اتصال به کیف‌پول کاربر یک فایل به نام connect.js ساخته شد، تمامی اعمال ارتباطی با کیف پول کاربر به عنوان چند تابع در این فایل جمع آوری شده‌اند، این فایل به صورت یک آداپتور میان Web3JS و کد کاپو عمل می‌کند. تمامی قابلیت‌های مورد نیاز مانند اتصال به کیف‌پول و ورود[۸] کاربر، خروج[۹] کاربر، گرفتن آدرس و شبکه‌ی کیف پول و ... در این فایل انجام می‌شود.

فرانت‌اند کاپو پس از تایید کاربر و دریافت آدرس کیف‌پول او، آن را در sessionStorage ذخیره می‌کند، از این طریق متوجه می شود که آیا کاربر وارد شده است یا خیر و با چه آدرسی. کاپو پیش از اتصال به کیف پول کاربر چک می‌کند که کیف پول روی شبکه یکسانی با شبکه فعلی کاپو باشد و در غیر این صورت به کاربر هشدار می‌دهد. همچنین در فرانت‌اند کاپو برای داشتن تجربه کاربری بهتر تلاش شده است. نکاتی مانند عدم نمایش قابلیت‌هایی مانند ساخت و ارسال توکن هنگامی که کیف پول کاربر به اپلیکیشن متصل نیست، جابه‌جایی آسان میان صفحات به کمک react-router، طراحی responsive برای رایانه و گوشی موبایل، نمایش alert ها و error های مناسب به کاربر، نمایش loading هنگامی که تراکنش‌ها در حالت pending هستند و نمایش پیام‌های مناسب با توجه به نتیجه تراکنش‌های کاربر.

[8] Login
[9] Logout

برای این که کاربرها بتوانند با قراردادهوشمند ارتباط برقرار کنند نیاز است که فرانت‌اند اپلیکیشن در سروری بارگذاری شود. خوشبختانه گیت‌هاب قابلیت به نام Github Pages در اختیار کاربرانش قرار می‌دهد که به کمک آن می‌توان فرانت‌اند اپلیکیشن را در آدرسی متناسب با آدرس مخزن کد در گیت‌هاب بارگذاری کرد و کاربران با رجوع به آن آدرس می‌توانند فرانت‌اند اپلیکیشن را ببینند و از آن استفاده کنند.

این قابلیت گیت‌هاب در واقع به این صورت عمل می‌کند که یک برنچ به نام gh-pages در repository پروژه می‌سازد و هربار که دستور دیپلوی پروژه توسط گیت‌هاب اجرا می‌شود، یک بیلد از پروژه می‌گیرد و فایل‌های خروجی بیلد روی این برنچ پوش می‌شوند. سپس این فایل‌ها روی آدرسی متناسب با آدرس repository دیپلوی می‌شوند. برای مثال آدرس ریپازیتوری و فرانت‌اند اپلیکیشن کاپو به صورت زیر است:

- آدرس ریپازیتوری: `https://github.com/bshramin/cappu`
- آدرس فرانت‌اند: `https://bshramin.github.io/cappu`

البته دیپلوی شدن فرانت‌اند روی Github Pages با ایجاد مشکلاتی در routing همراه بود که رفع شدند.

۵.۴ داکرایز شدن، پایپلاین‌ها وگیت

اقدامات زیر به منظور سرعت بخشیدن و تسهیل فرآیندهای توسعه و دیپلوی انجام شدند.

۱.۵.۴ داکرایز شدن تست‌های قراردادهوشمند

برای سرعت بخشیدن به توسعه قراردادهوشمند، این نیازمندی به وجود آمد که بعد از پوش شدن هر تغییر روی گیت‌هاب تست‌های قرارداد به صورت خودکار اجرا شوند. به این منظور پیش از هر چیز تست‌های قراردادهوشمند باید بتوانند به صورت داکرایز اجرا شوند.

برای داکرایز کردن اجرای تست‌های قرارداد هوشمند، اول سعی در این بود که یک ایمیج داکر پایه که ترافل روی آن نصب شده باشد پیدا شود، اما نسخه ترافل نمونه‌هایی که یافت شد با نسخه مورد نظر همخوانی نداشت.

در نتیجه یک ایمیج پایه داکر نوشته شد که داکرفایل آن را می‌توان در گیت‌هاب [10] مشاهده کرد، همچنین این ایمیج داکر در داکرهاب [11] نیز پوش شد.

سپس داکرفایل دیگری نوشته شد که با استفاده از این ایمیج پایه تست‌های قرارداد را اجرا کند. تست‌های قرارداد در این ایمیج که ترافل بر روی آن نصب شده است با اجرای دستور truffle test اجرا می‌شود.

۲.۵.۴ اجرای خودکار تست‌های قرارداد

با داشتن داکرفایلی که با بیلد و اجرای آن تست‌های قراردادهوشمند اجرا می‌شوند، تست‌های قراردادهوشمند می‌توانند به عنوان یکی از مراحل پایپلاین پروژه در گیت‌هاب نیز اجرا گردند. به این صورت در هر مرج ریکوئست به برنچ master و با پوش شدن یک کامیت در برنچ master تست‌ها به صورت خودکار در پایپلاین گیت‌هاب اجرا می‌شوند. به این ترتیب سرعت توسعه و اطمینان از کدهای قرارداد بیشتر می‌شود.

۳.۵.۴ دیپلوی خودکار فرانت‌اند

برای ساده‌سازی بیشتر فرآیند دیپلوی فرانت‌اند و سرعت بخشیدن به توسعه آن، این قابلیت پیاده سازی می‌شود که پس از هربار ایجاد تغییر در فرانت‌اند، به جای این که توسعه‌دهنده با اجرای دستوراتی فرانت‌اند را به کمک صفحات گیت‌هاب دیپلوی کند، فرانت‌اند پس از پوش شدن تغییرات جدید روی برنچ اصلی ریپازیتوری دیپلوی می‌شود.

برای پیاده‌سازی این قابلیت از Github Actions که در واقع پایپلاین‌های گیت‌هاب برای یک پروژه هستند استفاده می‌شود. تنها نکته‌ای که باید به آن توجه شود این است که این استیج از پایپلاین یک تفاوت اصلی با استیج‌های دیگر دارد. استیج‌های دیگر فقط می‌خواهند که کدهای ریپازیتوری را بخوانند و نمی‌خواهند چیزی را در ریپازیتوری تغییر دهند، اما این استیج می‌خواهد که کدهای فرانت‌اند را بیلد کند و سپس فایل‌های بیلد شده را روی برنچ دیگری به نام gh-pages پوش کند. پس این استیج پایپلاین نیاز به دسترسی پوش کردن کد روی ریپازیتوری دارد.

[10] https://github.com/bshramin/truffle-docker
[11] https://hub.docker.com/r/aminbshr/truffle

برای پیاده‌سازی این قابلیت به این صورت عمل می‌شود که نخست یک داکرفایل نوشته می‌شود که در آن کدهای فرانت‌اند بیلد و سپس به کمک صفحات گیت‌هاب روی برنچ gh-pages پوش و دیپلوی می‌شوند. اما این کانتینر برای این که بتواند کدها را روی ریپازیتوری پوش کند نیاز به یک توکن از گیت‌هاب دارد، به همین دلیل برای این داکرفایل یک ENV تعریف می‌شود و هنگامی که در استیج دیپلوی فرانت‌اند این داکرفایل بیلد و اجرا می‌شود توکنی که از گیت‌هاب گرفته شده است به عنوان env به این کانتینر داده می‌شود. به این ترتیب این توکن درون کانتینر داکر وجود خواهد داشت و Github Pages از آن استفاده خواهد کرد.

فصل ۵

بحث و نتیجه‌گیری

۱.۵ مقدمه

تاکنون شما در پایان‌نامه‌ای که مشغول نوشتن آن هستید، پاسخ چهار سؤال را داده‌اید:

- چرا تحقیق را انجام دادید؟ (مقدمه)
- دیگران در این زمینه چه کارهایی کرده‌اند و تمایز کار شما با آنها؟ (مرور ادبیات)
- چگونه تحقیق را انجام دادید؟ (روش‌ها)
- چه از تحقیق به دست آوردید؟ (یافته‌ها)

حال زمان آن فرا رسیده که با توجه به تمامی مطالب ذکر شده، در نهایت به سؤال آخر پاسخ دهید:

- چه برداشتی از یافته‌های تحقیق کردید؟ (نتیجه‌گیری)

در واقع در این بخش، هدف، پاسخ به این سوال است که چه برداشتی از یافته‌ها کردید و این یافته‌ها چه فایده‌ای دارند؟

نتیجه‌گیری مختصری بنویسید. ارائهٔ داده‌ها، نتایج و یافته‌ها در فصل چهارم ارائه می‌شود. در این فصل تفاوت، تضاد یا تطابق بین نتایج تحقیق با نتایج دیگر محققان باید ذکر شود. تفسیر و تحلیل نتایج نباید بر اساس

حدس و گمان باشد، بلکه باید **برمبنای نتایج عملی استخراج‌شده** از تحقیق و یا **استناد به تحقیقات دیگران** باشد. با توجه به حجم و ماهیت تحقیق و با صلاحدید استاد راهنما، این فصل می‌تواند تحت عنوانی دیگر بیاید یا به دو فصل جداگانه با عناوین مناسب، تفکیک شود. این فصل فقط باید به جمع‌بندی دست‌آوردهای فصل‌های سوم و چهارم محدود و از ذکر موارد جدید در آن خودداری شود. در عنوان این فصل، به جای کلمهٔ «تفسیر» می‌توان از واژگان «بحث» و «تحلیل» هم استفاده کرد. این فصل شاید مهم‌ترین فصل پایان‌نامه باشد.

در این فصل خلاصه‌ای از یافته‌های تحقیق جاری ارائه می‌شود. این فصل می‌تواند حاوی یک مقدمه، شامل مروری اجمالی بر مراحل انجام تحقیق باشد (حدود یک صفحه). مطالب پاراگراف‌بندی شود و هر پاراگراف به یک موضوع مستقل اختصاص یابد. فقط به ارائهٔ یافته‌ها و دست‌آوردها بسنده شود و از تعمیم بی‌مورد نتایج خودداری شود. تا حد امکان از ارائهٔ جداول و نمودارها در این فصل اجتناب شود. از ارائهٔ عناوین کلی در حوزهٔ تحقیق و قسمت پیشنهاد تحقیقات آتی خودداری شود و کاملاً در چارچوب و زمینهٔ مربوط به تحقیق جاری باشد. این فصل حدود ۱۰-۱۵ صفحه است.

۲.۵ محتوا

به ترتیب شامل موارد زیر است:

۱.۲.۵ جمع‌بندی

خلاصه‌ای از تمام یافته‌ها و دست‌آوردهای تحقیق جاری است.

۲.۲.۵ نوآوری

این قسمت، نوآوری تحقیق را بر اساس یافته‌های آن تشریح می‌کند. که دارای دو بخش اصلی است:

۱. نوآوری تئوری، یعنی تمایز تئوریک کار با کارهای محققین قبلی.

۲. نوآوری عملی، یعنی توصیه‌های محقق به صنعت برای بهبود بخشیدن به کارها، بر اساس یافته‌های تحقیق.

۳.۲.۵ پیشنهادها

این بخش، عناوین و موضوعات پیشنهادی را برای تحقیقات آتی، بیشتر در زمینهٔ مورد بحث در آینده ارائه می‌کند.

۴.۲.۵ محدودیت‌ها

در اینجا انواع محدودیت‌های تحقیق تشریح می‌شوند؛ از جمله، محدودیت‌هایی که کنترل آن از عهده محقق خارج است، مانند انتخاب نوع یافته‌ها؛ و همچنین دیگر محدودیت‌هایی که کنترل آن در دست محقق است، مانند موضوع و محل تحقیق و ... . تأثیر این محدودیت‌ها بر یافته‌های تحقیق در این قسمت شرح داده می‌شوند.

فصل ۶

آشنایی سریع با برخی دستورات لاتک

در این فصل ویژگی‌های مهم و پرکاربرد زی‌پرشین و لاتک معرفی می‌شود. برای راهنمایی بیشتر و به‌کاربردن ویژگی‌های پیشرفته‌تر به راهنمای زی‌پرشین و راهنمای لاتک مراجعه کنید. برای آگاهی از دستورات لاتک که این خروجی را تولید کرده‌اند فایل appendix1.tex را ملاحظه فرمایید. [۱]

۱.۶ بندها و زیرنویس‌ها

هر جایی از نوشتهٔ خود، اگر می‌خواهید به سر سطر بروید و یک بند (پاراگراف) تازه را آغاز کنید، باید یک خط را خالی بگذارید[۲] مانند این:

حالا که یک بند تازه آغاز شده است، یک زیرنویس انگلیسی[۳] هم می‌نویسیم!

[۱] بیشتر مطالب این بخش از مثال xepersian_example.tex گرفته شده‌اند که توسط آقای امیرمسعود پورموسی آماده شده است.
[۲] یعنی دوبار باید کلید Enter را بزنید.
[3] English Footnote!

۲.۶ فرمول‌های ریاضی

اینجا هم یک فرمول می‌آوریم که شماره دارد:

$$A = \frac{c}{d} + \frac{q^{۲}}{\sin(\omega t) + \Omega_{۱۲}} \tag{۱.۶}$$

در لاتک می‌توان به کمک فرمان {}\label به هر فرمول یک نام نسبت داد. در فرمول بالا نام eq:yek را برایش گذاشته‌ایم (پروندهٔ tex همراه با این مثال را ببینید). این نام ما را قادر می‌کند که بعداً بتوانیم با فرمان {eq:yek}\ref به آن فرمول با شماره ارجاع دهیم. یعنی بنویسیم فرمول ۱.۶. لاتک خودش شمارهٔ این فرمول‌ها را مدیریت می‌کند.[۴] این هم یک فرمول که شماره ندارد:

$$A = |\vec{a} \times \vec{b}| + \sum_{n=۰}^{\infty} C_{ij}$$

این هم عبارتی ریاضی مانند $\sqrt{a^{۲} + b^{۲}}$ که بین متن می‌آید.

۱.۲.۶ یک زیربخش

این زیربخش ۱.۲.۶ است؛ یعنی یک بخش درون بخش ۲.۶ است.

۱.۱.۲.۶ یک زیرزیربخش

این هم یک زیرزیربخش است. در لاتک می‌توانید بخش‌های تودرتو در نوشته‌تان تعریف کنید تا ساختار منطقی نوشته را به خوبی نشان دهید. می‌توانید به این بخش‌ها هم با شماره ارجاع دهید، مثلاً بخش فرمول‌های ریاضی شماره‌اش ۲.۶ است.

[۴] یعنی اگر بعداً فرمولی قبل از این فرمول بنویسیم، خودبه‌خود شمارهٔ این فرمول و شمارهٔ ارجاع‌ها به این فرمول یکی زیاد می‌شود. دیگر نگران شماره‌گذاری فرمول‌های خود نباشید!

۳.۶ نوشته‌های فارسی و انگلیسی مخلوط

نوشتن یک کلمهٔ انگلیسی بین متن فارسی بدیهی است، مانند Example در این جمله.[۵] نوشتن یک عبارت چندکلمه‌ای مانند More than one word کمی پیچیده‌تر است.

اگر ناگهان تصمیم بگیرید که یک بند کاملاً انگلیسی را بنویسید، باید:

This is an English paragraph from left to right. You can write as much as you want in it.

۴.۶ افزودن تصویر به نوشته

پروندهٔ تصویر دلخواه خود را در کنار پروندهٔ tex قرار دهید. سپس به روش زیر تصویر را در نوشتهٔ خود بیاورید:

```
\includegraphics{YourImageFileName}
```

به تصویرها هم مانند فرمول‌ها و بخش‌ها می‌توان با شماره ارجاع داد. مثلاً تصویر ۱.۶ یک شیر علاقه‌مند به لاتک را در حال دویدن نشان می‌دهد. برای جزئیات بیشتر دربارهٔ روش گذاشتن تصویرها در نوشته باید راهنماهای لاتک را بخوانید.

شکل ۱.۶: در این تصویر یک شیر علاقه‌مند به لاتک را در حال دویدن می‌بینید.

[۵] هرچند بهتر است باز هم آن کلمه را مانند Example در این جمله بنویسید.

به تصویرها هم مانند فرمول‌ها و بخش‌ها می‌توان با شماره ارجاع داد. مثلاً تصویر بالا شماره‌اش ۱.۶ است. برای جزئیات بیشتر دربارهٔ روش گذاشتن تصویرها در نوشته باید راهنماهای لاتک را بخوانید.

۵.۶ محیط‌های شمارش و نکات

برای فهرست‌کردن چندمورد، اگر ترتیب برایمان مهم نباشد:

- مورد یکم
- مورد دوم
- مورد سوم

و اگر ترتیب برایمان مهم باشد:

۱. مورد یکم
۲. مورد دوم
۳. مورد سوم

می‌توان موردهای تودرتو داشت:

۱. مورد ۱
۲. مورد ۲
   (آ) مورد ۱ از ۲
   (ب) مورد ۲ از ۲
   (ج) مورد ۳ از ۲
۳. مورد ۳

شماره‌گذاری این موردها را هم لاتک انجام می‌دهد.

۶.۶ تعریف و قضیه

برای ذکر تعریف، قضیه و مثال مثالهای ذیل را ببینید.

تعریف ۱.۶.۶. مجموعه همه ارزیابی‌های (پیوسته) روی (X, τ)، دامنه توانی احتمالی X نامیده می‌شود.

قضیه ۲.۶.۶ (باناخ-آلااغلو). اگر V یک همسایگی ۰ در فضای برداری توپولوژیکی X باشد و

$$K = \{\Lambda \in X^* : |\Lambda x| \leqslant ۱;\ \forall x \in V\}, \tag{۲.۶}$$

آنگاه K، ضعیف*-فشرده است که در آن، X^* دوگان فضای برداری توپولوژیکی X است به طوری که عناصر آن، تابعی‌های خطی پیوسته روی X هستند.

تساوی (۲.۶) یکی از مهم‌ترین تساوی‌ها در آنالیز تابعی است که در ادامه، به وفور از آن استفاده می‌شود.

مثال ۳.۶.۶. برای هر فضای مرتب، گردایه

$$U := \{U \in O : U = \uparrow U\}$$

از مجموعه‌های بالایی باز، یک توپولوژی تعریف می‌کند که از توپولوژی اصلی، درشت‌تر است.

حال تساوی

$$\sum_{n=۱}^{+\infty} ۳^n x + ۷x = \int_{۱}^{n} ۸nx + \exp\left(۲nx\right) \tag{۳.۶}$$

را در نظر بگیرید. با مقایسه تساوی (۳.۶) با تساوی (۲.۶) می‌توان نتیجه گرفت که ...

۷.۶ چگونگی نوشتن و ارجاع به مراجع

در لاتک به راحتی می‌توان مراجع خود را نوشت و به آنها ارجاع داد. به عنوان مثال برای معرفی کتاب گنزالس [؟] به عنوان یک مرجع می‌توان آنرا به صورت زیر معرفی نمود:

```
\bibitem{Gonzalez02book}
Gonzalez, R.C., and Woods, R.E. {\em Digital Image Processing}, 3rd ed..
Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.
```

در دستورات فوق Gonzalez02book برچسبی است که به این مرجع داده شده است و با استفاده از دستور `\cite{Gonzalez02book}` می‌توان به آن ارجاع داد؛ بدون این که شماره‌اش را در فهرست مراجع‌مان بدانیم.

اگر این اولین مرجع ما باشد در قسمت مراجع به صورت زیر خواهد آمد:

[1] Gonzalez, R.C., and Woods, R.E. *Digital Image Processing*, 3rd ed.. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

این شیوهٔ تعریف مراجع بسیار ابتدایی است و اگر فرمت مراجع، ترتیب یا تعداد آنها را خواسته باشید تغییر دهید، به عنوان مثال ابتدا حرف اول نام نویسنده بیاید و سپس نام خانوادگی، باید همه کارها را به صورت دستی انجام دهید! چون در یک پروژه/پایان‌نامه/رساله یا مقاله باید کنترل کاملی بر مراجع خود داشته باشید و به راحتی بتوانید قالب مراجع را عوض کنید، بنابراین می‌بایست از BibTEX استفاده کنید که درپیوست ۸ به آن پرداخته خواهد شد.

فصل ۷

جدول، نمودار و الگوریتم در لاتک

در این بخش نمونه مثالهایی از جدول، شکل، نمودار، الگوریتم و معادلات ریاضی را در لاتک خواهیم دید. دقت کنید که در پایان‌نامه‌ها و مقالات، باید قاعدهٔ «ارجاع به جلو[1]» رعایت شود؛ یعنی ابتدا در متن به شمارهٔ شکل، جدول یا معادله اشاره شود و بعد از آن (زیر آن) خود شکل، جدول یا معادله رسم شود. (توضیحات بیشتر در قسمت ۷.۷).

۱.۷ جدول

دستور اصلی برای رسم جدول در لاتک `tabular` می‌باشد که جدول (۱.۷) با استفاده از آن کشیده شده است؛ در `tabular` عرض جدول برابر با مجموع عرض ستون‌ها و حداکثر مساوی عرض متن است.

جدول ۱.۷: مدلهای تبدیل.

نام مدل	درجه آزادی	تبدیل مختصات	توضیح
انتقالی	۲	$x' = x + t_x$ $y' = y + t_y$	انتقال دوبعدی
اقلیدسی	۳	$x' = x\cos\theta - y\sin\theta + t_x$ $y' = x\sin\theta + y\cos\theta + t_y$	انتقالی+دوران

[1] Forward Referencing

برای اینکه عرض جدول قابل کنترل باشد، باید از دستورات `tabularx`، `tabulary` یا `tabu` استفاده کرد که راهنمای آنها در اینترنت وجود دارد. مثلاً جدول ۲.۷ با `tabularx` رسم شده که عرض جدول در آن ثابت بوده و ستون‌های از نوع `X` عرض خالی جدول را پر می‌کنند.

جدول ۲.۷: مدلهای تبدیل دیگر.

نام مدل	درجه آزادی	تبدیل مختصات	توضیح
مشابهت	۴	$x' = sx\cos\theta - sy\sin\theta + t_x$ $y' = sx\sin\theta + sy\cos\theta + t_y$	اقلیدسی+تغییر مقیاس
آفین	۶	$x' = a_{11}x + a_{12}y + t_x$ $y' = a_{21}x + a_{22}y + t_y$	مشابهت+اریب‌شدگی

۲.۷ معادلات ریاضی و ماتریس‌ها

تقریباً هر آنچه دانشجویان برای نوشتن فرمول‌های ریاضی لازم دارند، در کتاب mathmode آمده است. کافیست در خط فرمان، دستور زیر را وارد کنید:

```
texdoc mathmode
```

متن زیر شامل انواعی از اشیاء ریاضی است که با ملاحظه کدش می‌توانید با دستورات آن آشنا شوید.

شناخته‌شده‌ترین روش تخمین ماتریس هوموگرافی الگوریتم تبدیل خطی مستقیم (DLT[2]) است. فرض کنید چهار زوج نقطهٔ متناظر در دو تصویر در دست هستند، $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ و تبدیل با رابطهٔ $\mathbf{x}'_i = H\mathbf{x}_i$ نشان داده می‌شود که در آن:

$$\mathbf{x}'_i = (x'_i, y'_i, w'_i)^\top$$

و

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

[2]Direct Linear Transform

رابطه زیر را برای الگوریتم (۱.۷) لازم داریم.

$$\begin{bmatrix} \mathbf{0}^\top & -w'_i\mathbf{x}_i^\top & y'_i\mathbf{x}_i^\top \\ w'_i\mathbf{x}_i & \mathbf{0}^\top & -x'_i\mathbf{x}_i^\top \\ -y'_i\mathbf{x}_i^\top & x'_i\mathbf{x}_i^\top & \mathbf{0}^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \mathbf{0} \tag{۱.۷}$$

۳.۷ الگوریتم

۱.۳.۷ الگوریتم ساده با دستورهای فارسی

با مفروضات فوق، الگوریتم DLT به صورت نشان داده شده در الگوریتم (۱.۷) خواهد بود.

الگوریتم ۱.۷ الگوریتم DLT برای تخمین ماتریس هوموگرافی.

ورودی: $n \geq 4$ زوج نقطهٔ متناظر در دو تصویر $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$،

خروجی: ماتریس هوموگرافی H به نحوی‌که: $\mathbf{x}'_i = H\mathbf{x}_i$.

۱: برای هر زوج نقطهٔ متناظر $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ ماتریس $\mathbf{A}_i$ را با استفاده از رابطهٔ ۱.۷ محاسبه کنید.

۲: ماتریس‌های ۹ ستونی $\mathbf{A}_i$ را در قالب یک ماتریس $\mathbf{A}$ ۹ ستونی ترکیب کنید.

۳: تجزیهٔ مقادیر منفرد (SVD) ماتریس $\mathbf{A}$ را بدست آورید. بردار واحد متناظر با کمترین مقدار منفرد جواب $\mathbf{h}$ خواهد بود.

۴: ماتریس هوموگرافی H با تغییر شکل $\mathbf{h}$ حاصل خواهد شد.

۲.۳.۷ الگوریتم پیچیده و تودرتو با دستورهای فارسی

الگوریتم ۲.۷، یک الگوریتم ترکیبی و تودرتو است که با کمک دستورهای بستهٔ algorithmic نوشته شده است.

۳.۳.۷ الگوریتم با دستورهای لاتین

الگوریتم ۳.۷ یک الگوریتم با دستورهای لاتین است.

الگوریتم ۲.۷ الگوریتم اجرای برنامهٔ شبیه‌سازی

ورودی: زمان t_{max} به عنوان زمان لازم برای انجام شبیه سازی،
ورودی: گراف شبکه برای شبیه سازی،
خروجی: جدول تغییرات گراف از لحظهٔ ۰ تا t.
۱: **برای** تمام لحظات در بازهٔ ۰ تا t_{max} **انجام بده**
۲: **برای** تمام پیوندها **انجام بده**
۳: محاسبهٔ ضریب و نرخ انتقال پیوند
۴: محاسبهٔ کیفیت و نرخ یادگیری
۵: **پایان حلقهٔ برای**
۶: **برای** تمام گره‌ها **انجام بده**
۷: محاسبهٔ نرخ انتقال گره
۸: محاسبهٔ وضعیت جدید
۹: **پایان حلقهٔ برای**
۱۰: **اگر** تغییرات از مقدار δ کمتر است **آنگاه**
۱۱: شکستن حلقه {این شرط برای پایان قبل از رسیدن به محدودیت زمانی است، اگر تغییرات کمتر از δ باشد}
۱۲: **وگرنه اگر** زمان اجرای برنامه بیش از حد طول کشیده **و** $t > ۱۰۰$ **آنگاه**
۱۳: شکستن حلقه
۱۴: **پایان شرط اگر**
۱۵: **پایان حلقهٔ برای**
۱۶: **چاپ کن** زمان اجرای برنامه
۱۷: **بازگردان** ماتریس تغییرات زمانی

الگوریتم ۳.۷ الگوریتم RANSAC برای تخمین ماتریس هوموگرافی.

Require: $n \geq 4$ putative correspondences, number of estimations, N, distance threshold T_{dist}.

Ensure: Set of inliers and Homography matrix H.

1: **for** $k = 1$ to N **do**
2: Randomly choose 4 correspondence,
3: Check whether these points are colinear, if so, redo the above step
4: Compute the homography H_{curr} by DLT algorithm from the 4 points pairs,
5: ...
6: **end for**
7: Refinement: re-estimate H from all the inliers using the DLT algorithm.

۴.۷ کد

درج کد به زبان‌های مختلف به سادگی امکان‌پذیر است. برنامه ۱.۷ یک قطعه کد MATLAB را نشان می‌دهد.

```
% define a continuous function
f = '4*sin(2*pi*t)';
% plot a figure
ezplot(f);
```

برنامهٔ ۱.۷: نمونه کد MATLAB

۵.۷ تصویر

نمونهٔ یک تصویر را در فصل قبل دیدیم. دو تصویر شیر کنار هم را نیز در شکل ۱.۷ مشاهده می‌کنید.

(آ) شیر ۱ (ب) شیر ۲

شکل ۱.۷: دو شیر

۶.۷ نمودار

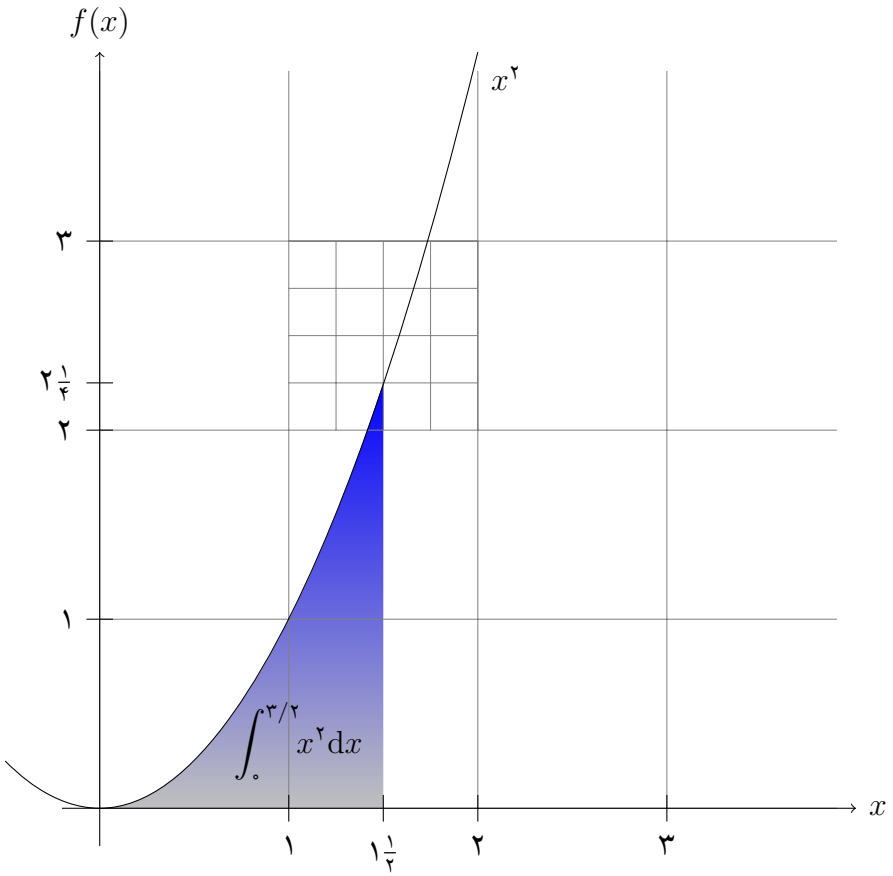
لاتک بسته‌هایی با قابلیت‌های زیاد برای رسم انواع مختلف نمودارها دارد. مانند بسته‌های Tikz و PSTricks. توضیح اینها فراتر از این پیوست کوچک است.[۳] یک نمودار رسم شده با بستهٔ TikZ در شکل ۲.۷ نشان داده شده است.

۷.۷ نحوه قرارگیری اشیای شناور

شکل‌ها، جداول و الگوریتم‌ها در لاتک اشیای شناور محسوب می‌شوند؛ یعنی خود لاتک تصمیم می‌گیرد آنها را در کجای صفحه ترسیم کند تا زیباتر باشد. اما می‌توان به لاتک توصیه کرد که آن را در قسمت خاصی از صفحه رسم کند. برای اینکه قاعدهٔ «ارجاع به جلو» رعایت شود باید فقط از پرچم [ht] استفاده کرد، که می‌گوید اگر جا شد شکل را دقیقاً در همین مکان و در غیراینصورت در بالای صفحه بعد رسم کن. بنابراین دستورات درج تصویر، جدول و الگوریتم به صورت زیر باید باشند:

```
\begin{figure/table/algorithm}[ht]
...
\end{figure/table/algorithm}
```

[۳] مثال‌هایی از بکارگیری بسته Tikz را می‌توانید در http://www.texample.net/tikz/examples/ ببینید. توصیه می‌شود دانشجویانی که قصد درج اشکالی مانند گراف را در سند خود دارند، مثالهایی از سایت مذکور را ملاحظه فرمایند.



شکل ۲.۷: یک نمودار زیبا با ارقام فارسی و قابلیت بزرگ‌نمایی بسیار، بدون از دست دادن کیفیت.

فصل ۸

مراجع، واژه‌نامه و حاشیه‌نویسی

۱.۸ مراجع و نقل‌قول‌ها

منابعِ پایان‌نامه، پایه و اساس تحقیق شما به حساب می‌آیند و ضرورت انجام مطالعه و روش‌های به کار رفته در بسیاری از قسمت‌های آن، به کمک منابع صورت می‌گیرد. در استفاده از مراجع علمی در پایان‌نامه، باید سعی کنید بیشتر از **منابع چاپ‌شده و مهم** استفاده کنید و ارجاع به داده‌های چاپ نشده، خلاصه‌ها و پایان‌نامه‌ها، سبب به‌هم‌خوردگی و کاهش اعتبار قسمت ارجاع منابع می‌شود. استفاده از منابع و نقل قول‌هایی به تحقیق شما ارزش می‌دهند که **در راستای هدف تحقیق بوده و به آن اعتبار ببخشند.** برخی از دانش‌جویان تصوّر می‌کنند که کثرت نقل‌قول‌ها و ارجاعات زیاد، مهم‌ترین معیار علمی شدن پایان‌نامه است؛ حال آنکه استناد به تعداد کثیری از منابع بدون مطالعه دقیق آنها و استفادهٔ مستقیم در پایان‌نامه، می‌تواند نشان‌دهندهٔ عدم احساس امنیت نویسنده باشد!

دو روش برای استفاده از نتایج، جملات، داده‌ها و روش‌های دیگران وجود دارد. یکی نقل‌قول مستقیم و دقیق است و دیگری استفاده غیرمستقیم در متن مقاله، که در ادامه به قواعد این دو نوع نقل‌قول و ارجاع‌دهی اشاره می‌کنیم:

نقل‌قول مستقیم: نقل‌قول مستقیم باید دقیق و بدون هیچ تغییری در جملات باشد. بهتر است این‌گونه نقل‌قول‌ها تا حد امکان کوتاه باشد. جملات کوتاه داخل گیومه قرار می‌گیرند و باید به منبع دقیق آن، طبق روش ارجاع‌دهی به منابع، اشاره شود. به عنوان مثال در [؟] آمده است که:

«با استفاده از فیلد AUTHORFA می‌توان معادل فارسی نام نویسندگان مقالات لاتین را در متن داشت. معمولاً در اسناد فارسی خواسته می‌شود که پس از ذکر معادل فارسی نام نویسنده، نام لاتین نویسنده(ها) به عنوان پاورقی درج شود [؟].»

نقل‌قول غیرمستقیم: نقل‌قول غیرمستقیم به معنی استفاده از ایده‌ها، نتایج، روش‌ها و داده‌های دیگران در درون متنِ پایان‌نامه، ولی به سبک خودتان و متناسب و هماهنگ با روند پایان‌نامهٔ شماست. در این حالت نیز باید متناسب با شیوهٔ ارجاع‌دهی به آن استناد شود.

با توجه به وجود سبک‌های مختلف ارجاع‌دهی، باید **روش قابل قبول و یکسانی** در طول پایان‌نامه برای اشاره به مراجع در متن و همچنین تهیه فهرست مراجع در انتهای پایان‌نامه بکار رود. مثلاً برای پایان‌نامه‌های مهندسی می‌توان از سبک ارجاع‌دهی IEEE[1] یا acm استفاده کرد. طبیعتاً باید تناظر یک‌به‌یک بین فهرست مراجع در انتهای گزارش و مراجع مورد استفاده در متن باشد[2].

برای سهولت مدیریت مراجعِ پروژه/پایان‌نامه/رساله ، اکیداً توصیه می‌شود از یک ابزار «مدیریت منابع» (با خروجی BibTEX) همچون Mendeley، Zotero، EndNote یا Citavi استفاده کنید.

۱.۱.۸ مدیریت مراجع با BibTEX

در بخش ۷.۶ اشاره شد که با دستور bibitem\ می‌توان یک مرجع را تعریف نمود و با فرمان cite\ به آن ارجاع داد. این روش برای تعداد مراجع زیاد و تغییرات آنها مناسب نیست. برای مدیریت منابع زیاد، سه بستهٔ BibTeX (پیش‌فرض)، natbib (ارجاع‌دهی در متن به صورت نویسنده–سال) و BibLaTeX (جدید و منعطف‌پذیر) وجود دارند. در ادامه توضیحاتی در مورد مدیریت منابع با BibTeX و natbib در زی‌پرشین خواهیم آورد که همراه با توزیع‌های معروف تِک عرضه می‌شوند [3].

یکی از روش‌های قدرتمند و انعطاف‌پذیر برای نوشتن مراجعِ مقالات و مدیریت مراجع در لاتک، استفاده از BibTeX است. روش کار با بیب‌تک به این صورت است که مجموعهٔ همهٔ مراجعی را که در پروژه/پایان‌نامه/رساله استفاده کرده یا خواهیم کرد، در پروندهٔ جداگانه‌ای با پسوند bib نوشته و به آن فایل در سند خودمان به صورت

[1] http://www.ieee.org/documents/ieeecitationref.pdf

[2] البته گاهی ممکن است محقق مرجعی را مورد مطالعه قرار داده لیکن در متن به آن اشاره نکرده باشد؛ برخی معتقدند در این موارد نیز آوردن آن در فهرست مراجع، اشکالی ندارد، به این شرط که از عنوان «فهرست منابع» به جای «فهرست مراجع» استفاده شود.

[3] روش BibLaTeX هنوز برای متون فارسی به درستی ترجمه نشده است.

مناسب لینک می‌دهیم. کنفرانس‌ها یا مجله‌های گوناگون برای نوشتن مراجع، قالب‌ها یا قراردادهای متفاوتی دارند که به آنها استیل‌های مراجع گفته می‌شود. در این حالت به کمک استیل‌های بیب‌تک خواهید توانست تنها با تغییر یک پارامتر در پروندهٔ ورودی خود، مراجع را مطابق قالب موردنظر تنظیم کنید. بیشتر مجلات و کنفرانس‌های معتبر یک فایل سبک (BibTeX Style) با پسوند bst در وب‌گاه خود می‌گذارند که برای همین منظور طراحی شده است.

به جز نوشتن مقالات، این سبک‌ها کمک بسیار خوبی برای تهیهٔ مستندات علمی همچون پایان‌نامه‌هاست که فرد می‌تواند هر قسمت از کارش را که نوشت مراجع مربوطه را به بانک مراجع خود اضافه نماید. با داشتن چنین بانکی از مراجع، وی خواهد توانست به راحتی یک یا چند ارجاع به مراجع و یا یک یا چند بخش را حذف یا اضافه نماید؛ مراجع به صورت خودکار مرتب شده و **فقط مراجع ارجاع داده شده در قسمت کتاب‌نامه خواهند آمد.** قالب مراجع به صورت یکدست مطابق سبک داده شده بوده و نیازی نیست که کاربر درگیر قالب‌دهی به مراجع باشد.

۲.۱.۸ سبک‌های مورد تأیید دانشگاه تهران

طبق «دستورالعمل نگارش و تدوین پایان‌نامه» دانشگاه تهران در [؟]، ارجاع در متن می‌تواند مطابق با هر یک از دو الگوی هاروارد یا ونکوور باشد:

سیستم نویسنده-سال (هاروارد): ذکر نام نویسنده و سال نشر در متن. در این الگو مراجع بر اساس حروف الفبا تنظیم می‌گردند.

سیستم شماره‌دار (ونکوور): ارجاع به مراجع به کمک شماره در متن. در این الگو شماره هر مرجع به ترتیب ظاهر شدن آن در متن در داخل کروشه قرار می‌گیرد. فهرست مراجع نیز بر اساس شماره مرجع (نه حروف الفبا) تنظیم می‌گردد.

در مدیریت منابع با **BibTeX**، ارجاع‌ها در متن تنها به شکل **شماره‌دار (ونکوور)** امکان‌پذیر است، گرچه فهرست مراجع می‌تواند با روش‌های مختلف مرتب شود. اگر بخواهیم ارجاع‌ها در متن به صورت **نویسنده-سال (هاروارد)** باشد باید از بستهٔ **natbib**[۴] و استیل‌های مختلف آن استفاده کنیم.

هنگام استفاده از روش نویسنده-سال نوع پرانتزگذاری‌ها در وسط و انتهای جمله با هم فرق خواهد داشت. به مثال زیر مطابق با دستورالعمل [؟] توجه کنید:

[4] Natural Sciences Citations & References

ابتدا\[؟] بستهٔ زی‌پرشین را برای حروف‌چینی فارسی اختراع کرد. بعدها سبک‌های ارجاع‌دهی فارسی و قالب‌های پایان‌نامه نیز مبتنی بر آن ساخته شد[؟]. ارجاع‌دهی به مراجع لاتین نیز در زی‌پرشین امکان‌پذیر است. مثلاً[؟] یک کتاب انگلیسی است و به راحتی به مقالات انگلیسی نیز می‌توان ارجاع داد[؟].

در این مثال، ۴ ارجاع در وسط و انتهای جمله به مراجع فارسی و انگلیسی آمده است. وقتی از سیستم نویسنده-سال استفاده می‌کنید، بهتر است ارجاع‌های آخر جمله کلاً داخل پرانتز بیاید؛ بدین منظور باید به جای \cite از \citep استفاده کنید. اما در سیستم شماره‌دار چون تمام ارجاع‌ها داخل کروشه می‌آیند این امر اهمیت ندارد.

نمی‌توانید در متن فارسی، اسم لاتین محقق خارجی را بیاورید و برای جلوگیری از ایجاد ابهام، صرف‌نظر از نام لاتین هم مجاز نیست! توصیه می‌شود که نام محقق خارجی در متن با حروف فارسی و در پاورقی اسم تمام نویسندگان به صورت انگلیسی آورده شود. نحوهٔ رعایت این نکته را می‌توانید در کد مثال بالا ببینید.

گرچه در تمپلت ورد [؟]، به صراحت ذکر شده که بهتر است برای پایان‌نامه‌های مهندسی از سبک IEEE استفاده شود (که از سیستم ونکوور تبعیت می‌کند)، اما ترتیب فهرست مراجع در IEEE بر اساس ترتیب ارجاع در متن بوده و مراجع انگلیسی و فارسی از هم تفکیک نمی‌شوند که متضاد با دستورالعمل [؟] و نیز متضاد عرف اکثر پایان‌نامه‌های فارسی است. بنابراین دقیقاً نمی‌توان سبک خاصی را برای مراجع پایان‌نامه‌های دانشگاه تهران اجبار کرد. مهم این است که **سبک ارجاع‌دهی در تمام طول یک کتابچه** (مثلاً پایان‌نامه، مقالات یک مجله یا کل یک کتاب) یکسان باشد. بهتر است **بسته به حوزه پایان‌نامه**، در این مورد با استاد راهنمای خود مشورت کنید.

۳.۱.۸ سبک‌های فارسی قابل استفاده در زی‌پرشین

تعدادی از سبک‌های فارسی بسته Persian-bib[۵] که برای زی‌پرشین آماده شده‌اند، عبارتند از:

- سبک‌های شماره‌دار:

unsrt-fa.bst این سبک متناظر با unsrt.bst می‌باشد. مراجع به ترتیب ارجاع در متن ظاهر می‌شوند.

plain-fa.bst این سبک متناظر با plain.bst می‌باشد. مراجع بر اساس نام‌خانوادگی نویسندگان، به ترتیب صعودی مرتب می‌شوند. همچنین ابتدا مراجع فارسی و سپس مراجع انگلیسی خواهند آمد.

[۵] برای اطلاع بیشتر به راهنمای بستهٔ Persian-bib مراجعه فرمایید.

acm-fa.bst این سبک متناظر با acm.bst می‌باشد. شبیه plain-fa.bst است. قالب مراجع کمی متفاوت است. اسامی نویسندگان انگلیسی با حروف بزرگ انگلیسی نمایش داده می‌شوند. (مراجع مرتب می‌شوند)

ieeetr-fa.bst این سبک متناظر با ieeetr.bst می‌باشد. (مراجع مرتب نمی‌شوند)

- سبک‌های نویسنده-سال:

plainnat-fa.bst این سبک متناظر با plainnat.bst می‌باشد. نیاز به بستهٔ natbib دارد. (مراجع مرتب می‌شوند)

chicago-fa.bst این سبک متناظر با chicago.bst می‌باشد. نیاز به بستهٔ natbib دارد. (مراجع مرتب می‌شوند)

asa-fa.bst این سبک متناظر با asa.bst می‌باشد. نیاز به بستهٔ natbib دارد. (مراجع مرتب می‌شوند)

با استفاده از استیل‌های فوق می‌توانید به انواع مختلفی از مراجع فارسی و لاتین ارجاع دهید. به عنوان مثال‌هایی از **مراجع انگلیسی**، مرجع [؟][۶] مقالهٔ یک ژورنال، مرجع [؟] مقالهٔ یک کنفرانس، مرجع [؟] یک کتاب، مرجع [؟] پایان‌نامهٔ کارشناسی ارشد و مرجع [؟] یک رسالهٔ دکتری می‌باشد.
همچنین در میان **مراجع فارسی**، مرجع [؟] مقالهٔ یک مجله، مرجع [؟] مقالهٔ یک کنفرانس، مرجع [؟] یک کتاب ترجمه‌شده با ذکر مترجمان و ویراستاران، مرجع [؟] پایان‌نامهٔ کارشناسی ارشد[۷]، مرجع [؟] یک رسالهٔ دکتری و مراجع [؟، ؟] نمونه‌های متفرقه هستند.

۴.۱.۸ ساختار فایل مراجع

برای استفاده از بیب‌تک باید مراجع خود را در یک فایل با پسوند bib ذخیره نمایید. یک فایل bib در واقع یک پایگاه داده از مراجع[۸] شماست که هر مرجع در آن به عنوان یک رکورد از این پایگاه داده با قالبی خاص ذخیره می‌شود. به هر رکورد یک مدخل[۹] گفته می‌شود. یک نمونه مدخل برای معرفی کتاب Digital Image Processing در ادامه آمده است:

```
@BOOK{Gonzalez02image,
  AUTHOR      = {Gonzalez,, Rafael C. and Woods,, Richard E.},
  TITLE       = {Digital Image Processing},
```

[۶] چون فیلد authorfa برای این مرجع تعریف نشده در سبک نویسنده-سال با حروف لاتین به آن در متن ارجاع می‌شود که غلط است.
[۷] همان‌طور که در بخش ۱.۸ اشاره شد، بهتر است زیاد از پایان‌نامه‌ها در مراجع استفاده نکنید.
[8] Bibliography Database
[9] Entry

```
  PUBLISHER  = {Prentice-Hall, Inc.},
  YEAR       = {2006},
  ISBN       = {013168728X},
  EDITION    = {3rd},
  ADDRESS    = {Upper Saddle River, NJ, USA}
}
```

در مثال فوق، @BOOK مشخصهٔ شروع یک مدخل مربوط به یک کتاب و Gonzalez02book برچسبی است که به این مرجع منتسب شده است. این برچسب بایستی یکتا باشد. برای آنکه بتوان **برچسب مراجع** را به راحتی به خاطر سپرد و حتی‌الامکان برچسب‌ها متفاوت با هم باشند، معمولاً از قوانین خاصی به این منظور استفاده می‌شود. یک قانون می‌تواند **فامیل نویسنده اول + دورقم سال نشر + اولین کلمهٔ عنوان اثر** باشد. به AUTHOR، TITLE، ... و ADDRESS فیلدهای این مدخل گفته می‌شود، که هر یک با مقادیر مربوط به مرجع پر شده‌اند. ترتیب فیلدها مهم نیست.

انواع متنوعی از مدخل‌ها برای اقسام مختلف مراجع همچون کتاب، مقالهٔ کنفرانس و مقالهٔ ژورنال وجود دارد که برخی فیلدهای آنها با هم متفاوت است. نام فیلدها بیانگر نوع اطلاعات آن می‌باشد. مثالهای ذکر شده در فایل MyReferences.bib کمک خوبی برای شما خواهد بود. با استفاده از سبک‌های فارسی آماده شده، محتویات هر فیلد می‌تواند به فارسی نوشته شود؛ ترتیب مراجع و نحوهٔ چینش فیلدهای هر مرجع را سبک مورد استفاده مشخص خواهد کرد.

در فایل MyReferences.bib که همراه با این پروژه/پایان‌نامه/رساله هست، مثال‌های مختلفی از مراجع آمده‌اند که برای درج مراجع خود، تنها کافیست مراجع‌تان را جایگزین موارد مندرج در آن نمایید.

برای بسیاری از مقالات لاتین حتی لازم نیست که مدخل مربوط به آنرا خودتان بنویسید. با جستجوی **نام مقاله + کلمه bibtex** در اینترنت سایت‌های بسیاری همچون ACM و ScienceDirect را خواهید یافت که مدخل bibtex مربوط به مقاله شما را دارند و کافیست آنرا به انتهای فایل MyReferences.bib اضافه کنید.

۵.۱.۸ نحوه اجرای BibTEX

پس از قرار دادن مراجع خود، برای ساخت فایل خروجی می‌توانید دستور زیر را (در ترمینال یا از طریق Texmaker) اجرا کنید:[۱۰]

[۱۰] فایل latexmkrc باید در کنار main.tex وجود داشته باشد.

```
latexmk -bibtex -pdf main.tex
```

ابزار latexmk مراحل مختلف ساخت خروجی لاتک را به طور خودکار و بهینه انجام می‌دهد و هر بار فقط مراحلی را که لازم باشد تکرار می‌کند. روش دستی‌تر این است که یک بار XeLaTeX را روی سند خود اجرا نمایید، سپس bibtex و پس از آن هم ۲ بار XeLaTeX را. در TeXMaker کلید F11 و در TeXWorks هم گزینهٔ BibTeX از منوی Typeset، BibTeX را روی سند شما اجرا می‌کنند.

۲.۸ واژه‌نامه‌ها و فهرست اختصارات

واژه‌نامه[11] یا فرهنگ لغات، مجموعه‌ای از اصطلاحات و تعاریف خاص و فنی است که معمولاً در انتهای یک کتاب می‌آید. چون پایان‌نامه خود یک متن تخصصی بلند محسوب می‌شود، استفاده از فرهنگ لغات در انتهای آن به شدت توصیه می‌شود، خصوصاً که احتمال استفاده از لغات تخصصی لاتین در آن بالاست. واژه‌نامه‌هایی که در انتهای کتاب‌های انگلیسی می‌آیند معمولاً تک‌زبانه هستند و معنی یک اصطلاح تخصصی در آنها، عمدتاً به صورت یک توصیف[12] طولانی آورده می‌شود. اما چون در متون فارسی، آوردن لغات انگلیسی مجاز نیست و باید معادل فارسی آنها وارد شود، جهت رفع ابهام معمولاً واژه‌نامهٔ فارسی به انگلیسی (و برعکس) در انتهای کتاب درج شده و توصیف‌ها در صورت نیاز در متن آورده می‌شوند.

فهرست اختصارات[13] شامل نمادهای کوتاهی است که اغلب از حروف ابتدایی کلمات یک عبارت طولانی ساخته شده‌اند. با اینکه اختصارات با حروف (بزرگ) لاتین نوشته می‌شوند، اما چون کوتاهند استفاده از آنها در میان متن فارسی مجاز است. با این حال برای رفع ابهام، عرف است که فهرستی از آنها شامل معنی هر نماد، در کنار دیگر فهرست‌ها در ابتدای متن درج شود.

در این قالب پایان‌نامه، برای ساخت و مدیریت واژه‌نامه و فهرست اختصارات از بستهٔ پیشرفتهٔ glossaries با موتور واژه‌نامه‌سازی xindy استفاده می‌شود. تنظیمات بهینهٔ این بسته در فایل glossaries-settings.tex عبارتند از:

[11] Glossary
[12] Description
[13] Acronym

- قبل از درج واژه‌ها در متن، باید مدخل آنها با دستور زیر (ترجیحاً در فایل جدای words.tex) تعریف شود:

```
\newword{Label}{Word}{واژه‌ها}{واژه}
```

- قبل از وارد کردن علائم اختصاری در متن، باید مدخل آنها نیز (ترجیحاً در فایل acronyms.tex) به صورت زیر تعریف شود:

```
\newacronym{Label}{Acr}{معنی‌اختصار}
```

- جهت درج یک علامت اختصاری یا معادل یک واژه تخصصی، کافی است از دستور `\gls{Label}` در متن استفاده کنید. دستور `\glspl{Label}` نیز برای آوردن معادل یک لغت در حالت جمع ساخته شده است.

- هنگام اولین استفاده از یک معادل فارسی یا اختصار در متن، معادل انگلیسی یا معنی آن در پاورقی آورده می‌شود. در صورتی که هر یک از این پیش‌فرض‌ها را دوست ندارید با ویرایش فایل glossaries-settings.tex می‌توانید آن را تغییر دهید.

- در انتهای پایان‌نامه با دستور `\printglossary` فهرست کلمات استفاده‌شده به ترتیب الفبای فارسی (واژه‌نامه فارسی به انگلیسی) و الفبای انگلیسی (واژه‌نامه انگلیسی به فارسی) درج می‌شود.

به عنوان مثال، با مشاهدهٔ کد این نوشته، نحوهٔ درج معادل فارسی متغیر تصادفی[۱۴] را در متن مشاهده می‌کنید. در نمایش واژهٔ متغیر تصادفی برای بار دوم، معادل لاتین در پاورقی نمی‌آید. در مورد درج علائم اختصاری، مثلاً می‌توان به رابطهٔ F^{15} اشاره کرد.

۳.۸ حاشیه‌نویسی در نسخه پیش‌نویس

اصلاح و بازبینی چندین و چندبارهٔ یک پایان‌نامه یا مقاله، از معمول‌ترین امور در نگارش آن می‌باشد. فرض کنید دانشجو پایان‌نامه یا مقالهٔ خود را (کامل یا ناقص) نوشته و می‌خواهد نظر استاد راهنما، اعضای آزمایشگاه یا

[14] Random Variable

(N) [15]

دیگر متخصصین را در مورد آن جویا شود. به جز مشاورهٔ حضوری، تلفنی یا از طریق ایمیل، برای اظهارنظر دقیق بر نوشته، می‌توان از ابزارهای حاشیه‌نویسی در فایل PDF یا tex نیز استفاده کرد.

یک راهکار مناسب برای حاشیه‌نویسی در فایل tex، استفاده از بسته todonotes می‌باشد که آقای خلیقی به تازگی امکان استفاده از آن را برای فارسی‌زبانان نیز فراهم آورده‌اند. بدین منظور، هر جایی که خواستید نکته یا نکاتی را در حاشیه متن یادداشت کنید، کافی است دستور زیر را وارد نمایید:

```
\todo{NOTE}
```

مثلاً استاد راهنما می‌تواند از دانشجو بخواهد که در بخشی توضیح بیشتری دهد. استاد راهنما یا داور حتی می‌تواند محل پیشنهادی برای درج یک تصویر را نیز به راحتی برای دانشجو مشخص کند. یکی دیگر از امکانات این بسته آن است که می‌توان فهرست نکات را در ابتدای سند داشت. بسته todonotes امکانات بسیاری دارد که در راهنمای آن معرفی شده است و با اجرای دستور زیر در خط فرمان می‌توانید آنها را مشاهده کنید:

```
texdoc todonotes
```

دقت کنید که توضیحات حاشیه‌ای و فهرست کارهای باقیمانده (نکات)، **فقط در نسخه پیش‌نویس**[۱۶] قابل دیدن هستند و در نسخه نهایی، نمایش داده نخواهند شد. برای استفاده از حالت پیش‌نویس باید گزینه draft به دستور \documentclass در ابتدای فایل main.tex اضافه شود. هنگامی‌که سند شما در حالت پیش‌نویس باشد:

۱. هیچ یک از صفحات آغازین پایان‌نامه، تا فهرست مطالب نمایش داده نمی‌شود (به جز صفحه اول).

۲. روی صفحه اول عبارت «پیش‌نویس» به صورت درشت و کم‌رنگ نمایش داده می‌شود.

۳. فهرست نکات درج شده توسط todo، پس از فهرست اصلی و با عنوان «فهرست کارهای باقیمانده» نمایش داده می‌شود.

۴. شماره صفحاتی که به هر مرجع ارجاع داده شده است در بخش مراجع نمایش داده می‌شود [۱۷].

هر یک از موارد بالا تا زمانی که نسخه نهایی پروژه/پایان‌نامه/رساله نیاز نباشد بسیار مورد توجه و مفید واقع می‌شوند.

[16] Draft

[۱۷] اعمال گزینهٔ pagebackref برای بستهٔ hyperref.

واژه‌نامهٔ فارسی به انگلیسی

الف

اختصار . . . . . Acronym

پ

پیش‌نویس . . . . . Draft

ت

توصیف . . . . . Description

م

متغیر تصادفی . . . . . Random Variable

و

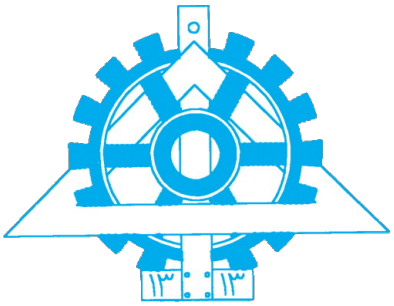
واژه‌نامه . . . . . Glossary

نمایه

Abstract

This thesis studies on writing projects, theses and dissertations using tehran-thesis class. It ...

Keywords Cappu, NFT, smart-contract, solidity, ERC721



University of Tehran
College of Engineering
Faculty of Electrical and Computer Engineering
Software Department

Cappu, a platform to mint and transfer data NFTs.

A Thesis submitted to the Graduate Studies Office
In partial fulfillment of the requirements for
The degree of Master of Science
in Computer Engineering - Software Engineering

By:

Amin Bashiri

Supervisor:

Prof. Ehsan Khamespanah

June 2022