

**دانشکده مهندسی کامپیوتر**

**گزارش پروژه کارشناسی**

**نرم‌افزار ترسیم نمودار مهندسی**

نگارش:

**محمد جواد رضوانیان**

استاد راهنما:

**دکتر علیرضا باقری**

استاد داور:

**دکتر مصطفی چهرقانی**

شهریور 1403

# 

# سپاسگزاری

در ابتدا از خداوند منان سپاسگزارم که همواره و در هر لحظه پشتیبان بندگان خود بوده و هر کجا که آنها در بن بست قرار گرفته‌اند، آنها را راهنمایی کرده. از خدایی سپاسگزارم که توفیق شاگردی نزد اساتید گرانقدر و بزرگواری را به من عطا کرد تا بتوان از ایشان در کنار درس‌های مرسوم امروزی، درس درست زندگی کردن و دغدغه‌مندی برای آینده‌ی کشورم را بیاموزم.

در ادامه از آقای دکتر علیرضا باقری استاد راهنمای پروژه و آقای دکتر صاحب‌الزمانی استاد مشاور عزیزم کمال تشکر و قدردانی را دارم که با راهنمایی‌ها و دلسوزی‌های این اساتید بزرگوار امکان تهیه این گزارش میسر شد.

محمد جواد رضوانیان

شهریور 03

# چکیده

نمودارها ابزارهای بصری قدرتمندی هستند که به مهندسان این امکان را می‌دهند تا داده‌ها، فرآیندها، و روابط پیچیده را به‌صورتی ساده و قابل فهم نمایش دهند. اهمیت نمودارها در مهندسی به دلیل توانایی آنها در بهبود درک مفاهیم فنی است. به کمک نمودارها، می‌توان اطلاعات فنی را به‌صورت گرافیکی ارائه داد، الگوهای داده‌ای را شناسایی کرد، و مشکلات را سریع‌تر تحلیل و حل کرد. نمودارها همچنین در مستندسازی پروژه‌ها، تسهیل تصمیم‌گیری و بهینه‌سازی طراحی‌ها نقش حیاتی ایفا می‌کنند، و به این ترتیب به افزایش دقت، کارایی، و نوآوری در فرآیندهای مهندسی منجر می‌شوند. مسئله اصلی استفاده از آن‌ها، پیچیده بودن ترسیم‌ها در لایه‌ی مهندسی است؛ اعداد و ارقام در این لایه بسیار زیاد و یا بزرگ بوده و یا برخی توابع و الگوها ترسیم‌شان به راحتی توسط انسان ممکن نیست. به طور کلی، نرم‌افزارهای ترسیم نمودارهای مهندسی به دلیل سرعت، دقت و قابلیت‌های پیشرفته‌شان به یک ابزار ضروری در هر پروژه مهندسی تبدیل شده‌اند.

با توجه به اهمیت این ابزار در رشته‌های مهندسی این سوال به وجود می‌آید که «آیا با استفاده از مفاهیم موجود در رشته‌ی مهندسی نرم‌افزار می‌توان یک نرم‌افزار استاندارد با استفاده از روش‌های چابک[[1]](#footnote-1) طراحی و پیاده‌سازی کرد که نیازهای تخصصی افراد را برطرف کند؟ هزینه تولید چنین نرم‌افزاری بدون استفاده از روش‌های مهندسی نرم‌افزار چه قدر اختلاف با این روش دارد و آیا لزومی دارد که این کار توسط یک تیم آموزش دیده انجام شود؟ در پروژه‌ی به این ابعاد آیا استفاده از روش‌های آزمون نرم‌افزار می‌تواند کارآمد باشد و کیفیت کار و میزان قابل اعتماد بودن نرم‌افزار را افزایش دهد؟»

برای پاسخ به این سوال ابتدا باید مفاهیم پایه‌ای برای تولید چنین نرم‌افزاری را بررسی کنیم و سپس با استفاده از روش چابک در هر چرخه کاری[[2]](#footnote-2)، فرایند توسعه و پیاده‌سازی را متناسب با نیاز مشتری - که یکی از شرکت‌های دانش‌بنیان است - انجام دهیم.

**واژه‌های کلیدی:** چابک، چرخه‌ی کاری، ترسیم، نگهداری اطلاعات، احراز هویت، نگهداری اطلاعات.

# فهرست مطالب

عنوان صفحه

[سپاسگزاری ت‌](#_Toc176555370)

[چکیده أ‌](#_Toc176555371)

[فهرست مطالب ب‌](#_Toc176555372)

[فهرست اشکال ث‌](#_Toc176555373)

[1. مقدمه 1](#_Toc176555376)

[1-1 مقدمه 2](#_Toc176555377)

[2. روش چابک 4](#_Toc176555379)

[1-2 تعریف روش چابک 5](#_Toc176555380)

[2-2 تاریخچه‌ای کوتاه 7](#_Toc176555381)

[3-2 عملکرد 9](#_Toc176555382)

[نتیجه گیری 11](#_Toc176555383)

[3. استخراج نیازمندی‌ها 12](#_Toc176555385)

[1-3. نمودارهای مورد استفاده 13](#_Toc176555386)

[2-3. طراحی نمونه‌ی کاغذی 13](#_Toc176555387)

[3-3. معماری انتزاعی سامانه 15](#_Toc176555388)

[4-3. مراحل انجام پروژه 15](#_Toc176555389)

[1-4-3. انتخاب زبان برنامه نویسی 15](#_Toc176555390)

[2-4-3. زمانبندی چرخه‌ها 19](#_Toc176555391)

[5-3 طراحی آزمون‌های TDD 19](#_Toc176555392)

[نتیجه‌گیری 19](#_Toc176555393)

[4. طراحی و پیاده‌سازی 20](#_Toc176555395)

[1-4 رابط کاربری 21](#_Toc176555396)

[2-4 پیاده‌سازی رابط کاربری 21](#_Toc176555397)

[**1-2-4 ویرایشگر متن** 21](#_Toc176555398)

[2-2-4 لیست فرمول‌ها 21](#_Toc176555399)

[3-2-4 روابط از پیش‌نوشته شده 21](#_Toc176555400)

[3-4 ساختمان داده‌ها 21](#_Toc176555401)

[نتیجه‌گیری 21](#_Toc176555402)

[تحلیل و بررسی میزان بهره وری 22](#_Toc176555404)

[1-5. ذخیره‌سازی اطلاعات 23](#_Toc176555405)

[2-5. بررسی آزمون‌های واحد 23](#_Toc176555406)

[نتیجه‌گیری 23](#_Toc176555407)

[نتیجه‌گیری و پیشنهادات 24](#_Toc176555409)

[نتیجه‌گیری 25](#_Toc176555410)

[پیشنهادات 25](#_Toc176555411)

[مراجع 26](#_Toc176555412)

# فهرست اشکال

## عنوان صفحه

[*شکل 1-1 میزان درآمد یک کارشناس حوزه هوش تجاری در شرکت‌های مختلف در سال 2021* **Error! Bookmark not defined.**](#_Toc175233657)

[*شکل 1-2 دید ابتدایی نسبت به هوش تجاری* **Error! Bookmark not defined.**](#_Toc175233658)

[*شکل 3-1 معماری اولیه هوش تجاری* **Error! Bookmark not defined.**](#_Toc175233659)

[*شکل 4-1 مثالی از الگوریتم RLE* **Error! Bookmark not defined.**](#_Toc175233660)

# **فصل اول**

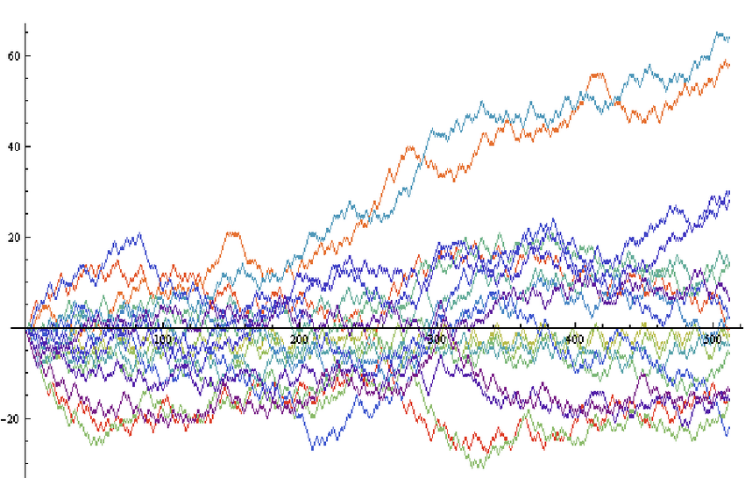
# 1. مقدمه

## 1-1 مقدمه

تاریخچه استفاده از نمودارها به دوران باستان بازمی‌گردد، زمانی که انسان‌ها برای درک و نمایش داده‌ها و مفاهیم مختلف به استفاده از تصاویر و نمادها روی آوردند. در طول قرون‌وسطی و رنسانس، استفاده از نمودارها به ویژه در زمینه‌های نجوم، جغرافیا، و ریاضیات پیشرفت چشمگیری داشت. با شروع انقلاب صنعتی در قرن هجدهم، نمودارها به ابزارهای مهمی در مهندسی، اقتصاد، و علوم تبدیل شدند. توسعه نمودارهای مدرن مانند نمودارهای خطی، ستونی، و دایره‌ای در قرن نوزدهم توسط افرادی مانند ویلیام هنری پلیفر[[3]](#footnote-3) به استانداردی در تجزیه و تحلیل داده‌ها و نمایش تصویری اطلاعات تبدیل شد. [1]

نمودارها در مهندسی نقش بی‌بدیلی دارند، زیرا امکان تجسم داده‌ها و تحلیل مفاهیم پیچیده را به‌صورت بصری فراهم می‌کنند. با استفاده از نمودارها، مهندسان قادرند تا الگوهای پنهان در داده‌ها را شناسایی کرده، روابط میان متغیرها را درک کنند و نتایج تحلیل‌ها را به‌طور موثر به دیگران منتقل نمایند. این ابزارهای گرافیکی، از طراحی سیستم‌های پیچیده تا مدیریت پروژه‌های بزرگ، در تمامی مراحل مهندسی کاربرد دارند. به‌ویژه در فرآیند طراحی و توسعه، نمودارها به مهندسان کمک می‌کنند تا ایده‌ها را به مدل‌های عملیاتی تبدیل کرده و از طریق شبیه‌سازی و تجزیه و تحلیل، بهینه‌سازی‌های لازم را انجام دهند. این امر منجر به افزایش دقت، کاهش خطا و تسهیل تصمیم‌گیری در پروژه‌های مهندسی می‌شود.

ترسیم نمودارها به‌وسیله انسان، به‌ویژه زمانی که با داده‌ها و سیستم‌های پیچیده سروکار داریم، می‌تواند فرآیندی چالش‌برانگیز و زمان‌بر باشد. دقت در ترسیم روابط پیچیده و نمایش صحیح داده‌ها نیازمند مهارت بالا و توجه دقیق به جزئیات است. هر گونه اشتباه در این فرآیند ممکن است به برداشت‌های نادرست و تصمیم‌گیری‌های اشتباه منجر شود. علاوه بر این، حجم بالای داده‌ها و نیاز به به‌روزرسانی مداوم نمودارها می‌تواند این کار را بسیار طاقت‌فرسا کند. از سوی دیگر، در برخی حوزه‌های مهندسی، نمودارها باید نه‌تنها دقیق بلکه کاملاً متناسب با استانداردهای تخصصی باشند که این امر نیاز به آشنایی کامل با قوانین و اصول طراحی دارد. به همین دلیل، ترسیم دستی نمودارها به‌خصوص در پروژه‌های بزرگ و پیچیده، علاوه بر خطاهای احتمالی، می‌تواند زمان و منابع زیادی را مصرف کند.



تصویر 1-1 نمونه ای از یک نمودار پیچیده

این نرم‌افزار با استفاده از متدولوژی چابک پیاده‌سازی شده است که باعث شده فرآیند توسعه آن به صورت تدریجی و انعطاف‌پذیر انجام گیرد. در این روش، پروژه به تعدادی چرخه یا چرخه کاری کوتاه تقسیم می‌شود که هر یک با هدف ارائه یک نسخه قابل‌استفاده از نرم‌افزار به پایان می‌رسد. این رویکرد امکان می‌دهد تا تیم توسعه به سرعت به بازخوردهای کاربران پاسخ دهد و به بهبود مداوم نرم‌افزار بپردازد. همچنین، استفاده از چابک تضمین می‌کند که نرم‌افزار با نیازهای متغیر کاربران و تغییرات محیط کسب‌وکار هماهنگ باشد و به‌روزرسانی‌های منظم و موثر انجام گیرد.

به منظور تضمین امنیت و محافظت از اطلاعات از الگوی احراز هویت در پروژه‌ استفاده شده است. احراز هویت به این معناست که تنها کاربران مجاز به دسترسی به روابط از پیش‌نوشته‌شده‌ی خود هستند و از دسترسی غیرمجاز آنها به روابط دیگران جلوگیری شود. این فرآیند نه تنها برای حفاظت از اطلاعات ضروری است، بلکه به ایجاد اعتماد میان کاربران و سیستم نیز کمک می‌کند.

هدف ما در این پروژه، ساخت یک نرم‌افزار ترسیم نمودار با استفاده از استانداردهای مهندسی نرم‌افزار و روش چابک است. اینکه بتوانیم با استفاده از چارچوب مطرح شده یک نرم‌افزار ترسیم نمودار طراحی و پیاده‌سازی کنیم و میزان هزینه‌ی آن را با زمانی که از این الگو پیروی نکنیم مقایسه کنیم.

در همین راستا در این گزارش، ابتدا بعد از تعریف مفاهیم اولیه‌ی به کار رفته از مهندسی نرم‌افزار در این پروژه، نگاهی اجمالی به ساختار چرخه‌های کاری می‌اندازیم. بعد از آن به بررسی هر یک از این چرخه‌ها پرداخته و در نهایت، عملکرد سیستم و بهره‌وری آن را بررسی می‌کنیم.

# **فصل دوم**

# 2. روش چابک

## 1-2 تعریف روش چابک

اصل چابک[[4]](#footnote-4) یک متدولوژی مدیریت پروژه و توسعه نرم‌افزار است که بر انعطاف‌پذیری، همکاری تیمی، و پاسخگویی سریع به تغییرات تأکید دارد. این رویکرد برخلاف روش‌های سنتی، که معمولاً بر یک برنامه‌ریزی جامع و تغییرناپذیر در ابتدای پروژه تکیه می‌کنند، اجازه می‌دهد تا پروژه‌ها به‌صورت تدریجی و از طریق چرخه‌های کوتاه و تکراری که به آن‌ها چرخه کاری[[5]](#footnote-5) گفته می‌شود، پیشرفت کنند. در هر چرخه، یک بخش مشخص از پروژه تکمیل می‌شود و محصولی قابل ارائه و استفاده به دست می‌آید که می‌تواند بازخوردهای کاربران را در خود جای دهد. این فرآیند به تیم‌ها امکان می‌دهد تا به سرعت به نیازها و اولویت‌های متغیر کاربران پاسخ دهند و نرم‌افزار یا محصول نهایی به‌طور مستمر بهبود یابد.

علاوه بر این، اصل چابک بر ارتباطات مستمر و نزدیک بین اعضای تیم و ذی‌نفعان تأکید دارد. در این رویکرد، جلسات منظم برای بازبینی عملکرد، ارزیابی پیشرفت، و تعیین اهداف جدید برگزار می‌شود. همچنین، تیم‌ها تشویق می‌شوند تا خودسازمان‌دهی داشته باشند و تصمیمات را به صورت مشارکتی و مبتنی بر بازخوردهای جمعی اتخاذ کنند. این شیوه کاری باعث می‌شود تا نه تنها محصول نهایی با کیفیت بالاتر و مطابق با نیازهای واقعی کاربران باشد، بلکه محیط کاری پویاتر و خلاقانه‌تری نیز ایجاد گردد. به طور کلی، اصل چابک با تأکید بر سرعت، انعطاف‌پذیری، و همکاری، به سازمان‌ها کمک می‌کند تا در مواجهه با تغییرات و چالش‌ها، کارآمدتر و موفق‌تر عمل کنند.

در جلسات روش چابک با ذی‌نفعان، چندین موضوع کلیدی مورد بحث قرار می‌گیرد که هدف اصلی آن‌ها اطمینان از هماهنگی، شفافیت و تطابق پروژه با نیازهای کاربران و کسب‌وکار است. این جلسات معمولاً به صورت منظم و در فواصل زمانی کوتاه، مانند پایان هر چرخه کاری برگزار می‌شوند و به عنوان بخش مهمی از فرآیند توسعه چابک شناخته می‌شوند. برخی از موضوعات اصلی که در این جلسات مورد بحث قرار می‌گیرد عبارتند از:

1. بررسی پیشرفت پروژه: تیم توسعه گزارش می‌دهد که در طول چرخه کاری گذشته چه میزان پیشرفت حاصل شده است و چه ویژگی‌ها یا عملکردهایی به نرم‌افزار اضافه شده‌اند. این بخش از جلسه به ذی‌نفعان کمک می‌کند تا وضعیت فعلی پروژه را به‌خوبی درک کنند.
2. پیش‌نمایش محصول: یکی از بخش‌های مهم جلسات با ذی‌نفعان، ارائه یک پیش‌نمایش زنده از قابلیت‌ها و ویژگی‌های جدید توسعه یافته در طول چرخه کاری است. این پیش‌نمایش محصول به ذی‌نفعان این امکان را می‌دهد تا عملکرد واقعی نرم‌افزار را ببینند و بازخورد فوری ارائه دهند. در ادامه برخی از روش های پیش‌نمایش را توضیح می‌دهیم
3. بازخورد و بررسی: ذی‌نفعان نظرات و پیشنهادات خود را درباره ویژگی‌های ارائه شده و جهت کلی پروژه بیان می‌کنند. این بازخوردها بسیار حیاتی هستند زیرا به تیم توسعه کمک می‌کنند تا در اسپرینت‌های بعدی به بهبود محصول و تطابق بهتر آن با نیازهای کاربران بپردازند.
4. بررسی مشکلات و چالش‌ها: هرگونه مشکلات یا چالش‌هایی که در طول چرخه پیش آمده است، با ذی‌نفعان در میان گذاشته می‌شود. این می‌تواند شامل موانع فنی، مشکلات تیمی، یا تغییرات غیرمنتظره در نیازهای پروژه باشد.
5. برنامه‌ریزی برای چرخه بعدی: در پایان جلسه، برنامه‌ها و اهداف چرخه بعدی با ذی‌نفعان به اشتراک گذاشته می‌شود. این شامل تعیین اولویت‌ها، ویژگی‌های مورد نظر، و زمان‌بندی است. هدف از این بخش اطمینان از هماهنگی کامل بین تیم توسعه و ذی‌نفعان در مورد مسیر پیش رو است.

به‌طور کلی، جلسات با ذی‌نفعان در روش چابک یک فرصت عالی برای ایجاد شفافیت، ارتباط نزدیک‌تر، و تضمین اینکه پروژه در راستای اهداف کسب‌وکار و نیازهای کاربران پیش می‌رود، فراهم می‌کند.

پیش‌نمایش[[6]](#footnote-6) یکی از مهم‌ترین بخش‌های جلسات چابک است که در آن تیم توسعه نتایج کار خود را به ذی‌نفعان و دیگر اعضای تیم ارائه می‌دهد. روش‌های مختلفی برای انجام پیش‌نمایش وجود دارد که بسته به نوع پروژه، محصول و مخاطبان می‌تواند متفاوت باشد. در زیر برخی از روش‌های رایج پیش‌نمایش را توضیح می‌دهم:

1. پیش‌نمایش زنده[[7]](#footnote-7)

- توضیح: در این روش، تیم توسعه محصول یا ویژگی‌های جدید را به صورت زنده و در محیط واقعی به ذی‌نفعان نمایش می‌دهد. این روش معمولاً بیشترین تأثیر را دارد زیرا ذی‌نفعان می‌توانند به‌صورت مستقیم عملکرد و کارایی محصول را مشاهده کنند.

- مزایا: شفافیت بالا، امکان دریافت بازخورد فوری، و نمایش محصول در حالت واقعی.

- چالش‌ها: نیازمند آماده‌سازی دقیق و بدون خطا بودن اجرای زنده است که ممکن است استرس‌زا باشد.

2. پیش‌نمایش ویدئویی[[8]](#footnote-8)

- توضیح: در این روش، یک ویدئو از عملکرد محصول یا ویژگی‌های جدید از پیش ضبط می‌شود و در جلسه پیش‌نمایش به نمایش گذاشته می‌شود. این روش به ویژه زمانی مفید است که نمایش زنده به دلیل محدودیت‌های فنی یا زمانی امکان‌پذیر نباشد.

- مزایا: امکان ویرایش و آماده‌سازی دقیق ویدئو، مناسب برای نمایش‌های پیچیده یا زمان‌بندی محدود.

- چالش‌ها: فقدان امکان تعامل زنده و دریافت بازخورد فوری.

3. پیش‌نمایش تعاملی[[9]](#footnote-9)

- توضیح: در این روش، به ذی‌نفعان اجازه داده می‌شود که خودشان با محصول یا ویژگی‌های جدید به صورت مستقیم تعامل داشته باشند. این نوع پیش‌نمایش معمولاً به‌صورت یک محیط تستی یا نسخه نمایشی محصول انجام می‌شود که کاربران می‌توانند آن را امتحان کنند.

- مزایا: تجربه کاربری واقعی، امکان دریافت بازخورد عملی از کاربران، و ارتقای اعتماد به محصول.

- چالش‌ها: نیاز به محیط تستی قابل‌اعتماد و زمان بیشتر برای اجرای دمو.

4. پیش‌نمایش اسلایدی[[10]](#footnote-10)

- توضیح: این روش شامل استفاده از اسلایدها برای نمایش و توضیح ویژگی‌ها و عملکردهای جدید است. این روش بیشتر بر روی توضیحات مفهومی و چگونگی کارکرد محصول تمرکز دارد و کمتر به نمایش زنده یا تعاملی می‌پردازد.

- مزایا: ساده برای آماده‌سازی، مناسب برای توضیح ویژگی‌های مفهومی یا نظری، و مدیریت زمان بهتر.

- چالش‌ها: ممکن است نتواند تجربه واقعی استفاده از محصول را به‌خوبی منتقل کند.

5. پیش‌نمایش از راه دور[[11]](#footnote-11)

- توضیح: این روش برای زمانی که ذی‌نفعان در مکان‌های مختلف جغرافیایی قرار دارند و امکان حضور فیزیکی در یک جلسه را ندارند، به کار می‌رود. معمولاً از طریق ابزارهای ویدئوکنفرانس یا اشتراک‌گذاری صفحه نمایش انجام می‌شود.

- مزایا: انعطاف‌پذیری مکانی، امکان ارائه به ذی‌نفعان در هر نقطه‌ای از جهان، و کاهش هزینه‌های سفر.

- چالش‌ها: وابستگی به کیفیت اینترنت و تجهیزات ارتباطی، و محدودیت‌های مربوط به تعامل زنده.

هر یک از این روش‌ها بسته به نیاز پروژه، سطح آمادگی محصول، و نوع ذی‌نفعان می‌تواند به صورت مستقل یا ترکیبی مورد استفاده قرار گیرد. انتخاب روش مناسب برای پیش‌نمایش می‌تواند تأثیر زیادی بر موفقیت جلسات و دریافت بازخوردهای ارزشمند داشته باشد.

## 2-2 تاریخچه‌ای کوتاه

تاریخچه روش چابک به‌عنوان یک متدولوژی نوین در توسعه نرم‌افزار و مدیریت پروژه، از نیاز به تغییر در رویکردهای سنتی و ناکارآمد در این حوزه‌ها سرچشمه می‌گیرد. برای درک کامل این تاریخچه، باید به چالش‌ها و محدودیت‌های روش‌های قدیمی در توسعه نرم‌افزار و چگونگی پیدایش و تکامل روش چابک بپردازیم.

پیش از ظهور روش چابک، بسیاری از پروژه‌های نرم‌افزاری بر اساس روش‌های سنتی مانند مدل آبشاری[[12]](#footnote-12) اجرا می‌شدند. در مدل آبشاری، فرآیند توسعه نرم‌افزار به مراحل مجزا و متوالی تقسیم می‌شود، از جمله نیازسنجی، طراحی، پیاده‌سازی، آزمایش و نگهداری. هر مرحله باید قبل از شروع مرحله بعدی به‌طور کامل تکمیل شود. این رویکرد به برنامه‌ریزی دقیق و جامع در ابتدای پروژه نیاز داشت و تغییرات در مراحل بعدی بسیار دشوار و پرهزینه بود.

با این حال، با گذشت زمان، مشکلات متعددی در اجرای این روش پدیدار شد. یکی از بزرگ‌ترین چالش‌ها این بود که نیازهای کاربران و بازار به سرعت تغییر می‌کرد و مدل آبشاری توانایی لازم برای پاسخگویی به این تغییرات را نداشت. علاوه بر این، پروژه‌ها اغلب با تأخیرهای زیاد و هزینه‌های بیشتر از حد پیش‌بینی مواجه می‌شدند، و در نهایت محصول نهایی ممکن بود با نیازهای واقعی کاربران تطابق نداشته باشد.

در دهه 1990، با توجه به مشکلات روش‌های سنتی، گروهی از متخصصان نرم‌افزار به دنبال راهکارهای جدیدی برای افزایش انعطاف‌پذیری و پاسخگویی در توسعه نرم‌افزار بودند. این تلاش‌ها به تدریج منجر به ظهور روش‌ها و چارچوب‌های جدیدی مانند اسکرام[[13]](#footnote-13) شد. این روش‌ها همگی به نوعی تلاش داشتند تا با ایجاد تغییرات کوچک و تدریجی در پروژه‌ها، به سرعت به نیازهای متغیر کاربران پاسخ دهند.

نقطه عطف واقعی در تاریخچه روش چابک در فوریه سال 2001 اتفاق افتاد، زمانی که 17 نفر از متخصصان برجسته نرم‌افزار، گرد هم آمدند تا در مورد روش‌های جدید توسعه نرم‌افزار بحث و گفتگو کنند. این گروه شامل افرادی بود که هر کدام تجربه قابل‌توجهی در توسعه نرم‌افزار داشتند.

نتیجه این نشست، تدوین بیانیه چابک[[14]](#footnote-14) بود که چهار ارزش اصلی و دوازده اصل را به‌عنوان مبانی توسعه چابک نرم‌افزار معرفی کرد. این ارزش‌ها عبارت بودند از:

1. افراد و تعاملات بیشتر از فرآیندها و ابزارها.

2. نرم‌افزار کارآمد بیشتر از مستندات جامع.

3. همکاری نزدیک با مشتری بیشتر از قراردادهای رسمی.

4. پاسخگویی به تغییرات بیشتر از پیروی از یک برنامه‌ی از پیش تعیین شده.

این بیانیه به سرعت مورد استقبال جامعه نرم‌افزاری قرار گرفت و به‌عنوان یک استاندارد برای توسعه چابک نرم‌افزار شناخته شد.

پس از انتشار بیانیه چابک، روش‌ها و چارچوب‌های مختلف چابک به سرعت در سازمان‌ها و تیم‌های توسعه نرم‌افزار مورد استفاده قرار گرفتند. اسکرام و XP از جمله اولین و پرکاربردترین روش‌های چابک بودند که به دلیل ساختار ساده و قابلیت اجرای آسان، به سرعت محبوبیت پیدا کردند.

با گذشت زمان، روش چابک از حوزه توسعه نرم‌افزار فراتر رفت و به دیگر حوزه‌های مدیریت پروژه نیز وارد شد. شرکت‌ها و سازمان‌های مختلف به مزایای این روش، از جمله کاهش زمان تحویل، افزایش رضایت مشتری، و بهبود کیفیت محصول، پی بردند و آن را به کار گرفتند.

با وجود موفقیت گسترده، روش چابک نیز با چالش‌ها و انتقاداتی مواجه شده است. برخی از این چالش‌ها شامل نیاز به تغییر فرهنگ سازمانی، پیچیدگی در مقیاس‌دهی به پروژه‌های بزرگ، و نیاز به همکاری نزدیک‌تر و مداوم بین تیم‌ها و ذی‌نفعان است. با این حال، با گذشت زمان و تکامل روش‌های مختلف چابک، بسیاری از این چالش‌ها نیز مورد توجه قرار گرفته و راه‌حل‌های مناسبی برای آن‌ها ارائه شده است.

امروزه، روش چابک به یکی از رایج‌ترین و پرکاربردترین متدولوژی‌های مدیریت پروژه و توسعه نرم‌افزار تبدیل شده است. بسیاری از سازمان‌ها در سراسر جهان از این روش برای مدیریت پروژه‌های خود استفاده می‌کنند و در تلاش هستند تا با تطبیق آن با نیازهای خاص خود، از مزایای چابک بهره‌مند شوند. روش چابک به‌طور مداوم در حال تکامل است و با افزودن چارچوب‌ها و ابزارهای جدید، همچنان به‌عنوان یک رویکرد پویا و کارآمد در مدیریت پروژه‌ها مورد توجه قرار دارد.

## 3-2 عملکرد

روش چابک بر اساس تکرارهای کوتاه‌مدت، همکاری نزدیک، و انعطاف‌پذیری در پاسخ به تغییرات بنا شده است. نحوه عملکرد این روش از طریق فرآیندهای مشخص، نقش‌های کلیدی، و ابزارهای گوناگون توضیح داده می‌شود. در ادامه، نحوه عملکرد روش چابک را به‌صورت مفصل توضیح خواهیم داد.

روش چابک بر اساس چهار ارزش اصلی و دوازده اصل کار می‌کند. این اصول به این معنی هستند که تیم‌های چابک باید:

- به تعاملات بین افراد بیشتر از فرآیندها و ابزارها اهمیت دهند.

- تمرکز بیشتری بر روی ارائه نرم‌افزار کارآمد نسبت به مستندات جامع داشته باشند.

- با مشتریان نزدیک‌تر همکاری کنند.

- به تغییرات به‌جای پیروی کورکورانه از برنامه‌های از پیش تعیین‌شده، به سرعت پاسخ دهند.

یکی از ویژگی‌های اصلی روش چابک، تقسیم پروژه به دوره‌های زمانی کوتاه و است. هر چرخه کاری معمولاً بین 1 تا 4 هفته طول می‌کشد و در این مدت، تیم توسعه بر روی یک مجموعه کوچک از ویژگی‌ها یا وظایف که به مجموعه چرخه[[15]](#footnote-15) معروف است، تمرکز می‌کند. در هر اسپرینت، هدف اصلی تولید یک محصول یا نسخه قابل ارائه[[16]](#footnote-16) است که قابل تست و ارزیابی باشد. چرخه‌ها به تیم‌ها امکان می‌دهند تا به‌صورت تکراری و تدریجی پیشرفت کنند و محصول نهایی را به تدریج بهبود بخشند.

روش چابک بر پایه برگزاری جلسات منظم برای هماهنگی و ارزیابی پیشرفت پروژه بنا شده است. این جلسات به‌طور معمول شامل موارد زیر هستند:

- جلسه برنامه‌ریزی چرخه کاری: در آغاز هر چرخه، تیم با کمک صاحب محصول (Product Owner) وظایف و اهداف آن چرخه کاری را تعیین می‌کند. در این جلسه، لیست نیازمندی‌های محصول بررسی می‌شود و تیم تصمیم می‌گیرد که چه وظایفی را در چرخه کاری پیش رو انجام دهد.

- جلسه روزانه (Daily Standup): هر روز، تیم یک جلسه کوتاه (معمولاً 15 دقیقه) برگزار می‌کند که در آن هر عضو تیم گزارش می‌دهد که چه کاری انجام داده، چه کاری خواهد کرد، و چه مشکلاتی ممکن است وجود داشته باشد. این جلسات به هماهنگی روزانه تیم کمک می‌کند و موانع را به سرعت شناسایی می‌کند.

- جلسه بازنگری چرخه کاری (Sprint Review): در پایان هر اسپرینت، تیم نسخه جدید محصول را به ذی‌نفعان و صاحب محصول ارائه می‌دهد. در این جلسه، تیم بازخوردهای فوری دریافت می‌کند و عملکرد خود را ارزیابی می‌کند.

- جلسه بازنگری عملکرد (Sprint Retrospective): پس از بازنگری اسپرینت، تیم در یک جلسه داخلی شرکت می‌کند تا فرآیندهای خود را بررسی کند و برای بهبود عملکرد در اسپرینت‌های آینده تصمیم‌گیری کند. این جلسه بر روی چگونگی بهبود فرآیندها و همکاری تیمی تمرکز دارد.

چندین نقش کلیدی در تیم‌های چابک وجود دارد که هر کدام مسئولیت‌های مشخصی دارند:

- صاحب محصول (Product Owner): مسئول تعریف و اولویت‌بندی ویژگی‌ها و نیازمندی‌های محصول است. صاحب محصول نماینده ذی‌نفعان و مشتریان است و باید اطمینان حاصل کند که تیم در مسیر تحقق اهداف کسب‌وکار قرار دارد.

- اسکرام مستر (Scrum Master): مسئول تسهیل فرآیندها و جلسات چابک است. اسکرام مستر به تیم کمک می‌کند تا بهترین عملکرد را داشته باشد و هرگونه مانعی که در راه است را برطرف می‌کند. اسکرام مستر همچنین از تطابق تیم با اصول چابک اطمینان حاصل می‌کند.

- تیم توسعه (Development Team): گروهی از افراد چندمهارته که به‌طور مشترک بر روی توسعه محصول کار می‌کنند. این تیم شامل برنامه‌نویسان، طراحان، تسترها و سایر افرادی است که برای تحویل نسخه‌های کاربردی محصول همکاری می‌کنند.

بک‌لاگ محصول (Product Backlog) لیستی از تمامی ویژگی‌ها، بهبودها، اصلاحات و نیازمندی‌هایی است که باید در محصول نهایی پیاده‌سازی شوند. این لیست توسط صاحب محصول نگهداری می‌شود و دائماً با توجه به بازخوردها و تغییرات به‌روزرسانی می‌شود.

بک‌لاگ چرخه کاری (Sprint Backlog) زیرمجموعه‌ای از بک‌لاگ محصول است که تیم توسعه تصمیم می‌گیرد در طول یک چرخه کاری خاص انجام دهد. این لیست معمولاً وظایف کوچک‌تر و دقیق‌تری را شامل می‌شود که تیم می‌تواند در مدت زمان کوتاه چرخه کاری تکمیل کند.

یکی از اصول مهم چابک، تحویل مستمر نسخه‌های قابل اجرا از محصول است. هر اسپرینت باید با یک نسخه جدید از محصول که قابل تست و استفاده است به پایان برسد. این رویکرد به تیم‌ها اجازه می‌دهد تا به سرعت بازخورد کاربران و ذی‌نفعان را دریافت کرده و در اسپرینت‌های بعدی بهبودهای لازم را اعمال کنند.

این بازخورد مداوم به اطمینان از اینکه محصول نهایی با نیازهای واقعی کاربران و تغییرات بازار تطابق دارد، کمک می‌کند. یکی از بزرگ‌ترین مزایای روش چابک، توانایی آن در مدیریت تغییرات است. در چابک، تغییرات به‌عنوان یک واقعیت اجتناب‌ناپذیر در فرآیند توسعه پذیرفته می‌شوند. تیم‌ها به‌جای مقابله با تغییرات، آن‌ها را به‌عنوان فرصتی برای بهبود محصول و رضایت مشتریان می‌بینند.

بک‌لاگ محصول و اولویت‌های آن به‌طور مداوم بر اساس بازخوردها و تغییرات محیطی به‌روزرسانی می‌شود. این انعطاف‌پذیری به تیم‌ها اجازه می‌دهد تا به سرعت به تغییرات پاسخ دهند و از انحراف پروژه از اهداف خود جلوگیری کنند.

روش چابک بر همکاری و ارتباطات نزدیک بین تمامی اعضای تیم و ذی‌نفعان تأکید دارد. این ارتباطات نزدیک به شفافیت در فرآیند توسعه، سرعت در شناسایی مشکلات، و بهبود کارایی تیم کمک می‌کند.

علاوه بر جلسات رسمی، تیم‌های چابک تشویق می‌شوند تا به‌صورت غیررسمی و در محیط کاری خود نیز به‌طور مستمر با هم در ارتباط باشند. این محیط همکاری باز به تیم‌ها اجازه می‌دهد تا سریع‌تر تصمیم‌گیری کنند و مشکلات را به‌صورت جمعی حل کنند.

روش چابک یک فرآیند ارزیابی و بهبود مستمر را دنبال می‌کند. تیم‌ها نه تنها به‌صورت مداوم عملکرد محصول را ارزیابی می‌کنند، بلکه فرآیندهای کاری خود را نیز مورد بازبینی قرار می‌دهند تا بهره‌وری و کیفیت کار را بهبود بخشند.

جلسات بازنگری اسپرینت و بازنگری عملکرد از جمله ابزارهایی هستند که به تیم‌ها کمک می‌کنند تا به این هدف دست یابند. تیم‌ها با تحلیل عملکرد گذشته، می‌توانند از تجربیات خود برای بهبود فرآیندهای آتی استفاده کنند.

## نتیجه گیری

روش چابک علی رغم روش‌های قبلی خود بسیار پویا بوده و به دلیل توجه بیشتر بین تعاملات افراد، سرعت انجام کار را بالا می‌برد. همچنین با ارائه‌ی نسخه‌های متعدد و در بازه‌های زمانی کوتاه، به ذی‌نفعان و مشتری این دلگرمی را می‌دهد که به محصول نهایی خود هر روز نزدیکتر می‌شوند. همچنین بدلیل جلسات کوتاه روزانه، کل تیم از عملکرد دیگران مطلع شده و هماهنگی بین اعضا را بالا می‌برد.

.

# **فصل سوم**

# 3. استخراج نیازمندی‌ها

## 1-3. نمودارهای مورد استفاده[[17]](#footnote-17)

یکی از اولین قدم‌های ساخت نرم‌افزار تشخیص کاربران سامانه است. به این منظور که چه کسی و برای حل چه مشکلی از سامانه ما استفاده خواهد کرد. یکی از معروف‌ترین راه حل‌های موجود استفاده از نمودارهای UML است. با استفاده از این نمودارها می‌توان یک نگاه کلی به استفاده کنندگان و نیازهای مربوط به آنها پیدا کرد. نمودار مربوط به سامانه ما مطابق با شکل زیر است.



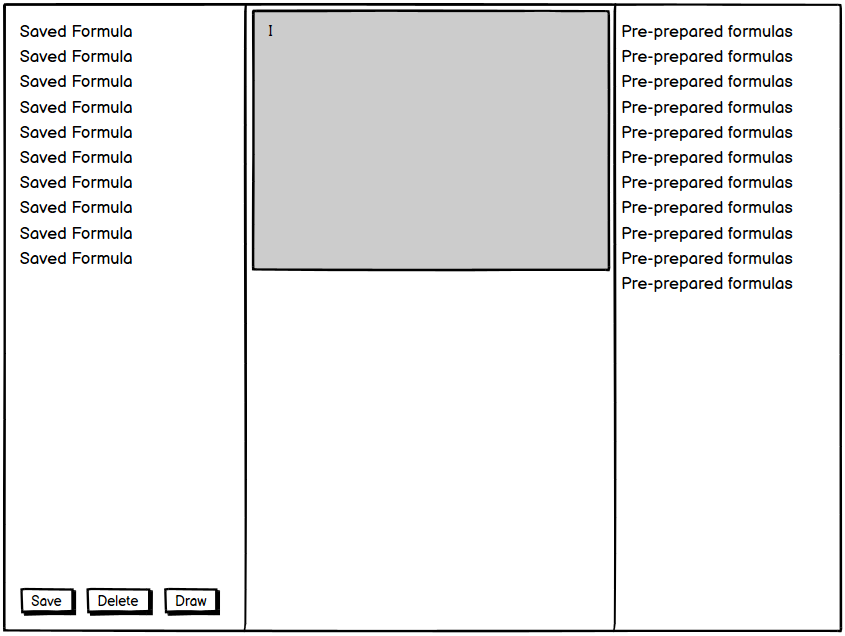
تصویر 2 نمودار استفاده کنندگان از سامانه

گروه‌های هدف در سامانه ما به سه گروه دانشمندان و تحلیلگران، اساتید و دانشجویان تقسیم می‌شوند. تحلیلگران نیاز دارند تا با تغییر دادن متغیرها اثر آنها را در نمودارهای خود ببینند تا با توجه به شبیه‌سازی و مدل سازی که از پیش انجام شده رفتار مدل خود را پیش بینی کنند. اساتید نیاز دارند تا برای تدریس بهتر و نمایش آنچه که دانشجویان باید از رفتار مدل بدانند، فرمول‌هایی را از پیش طراحی کرده و در اختیار دانشجویان خود قرار دهند. دانشجویان نیاز دارند تا به محض احراز هویت به فرمول‌های مربوط به خودشان دسترسی پیدا کنند.

در نتیجه ۵ عمل اصلی در سامانه ما انجام می‌شود؛ احراز هویت، نمایش فرمول‌های مربوط به خود، تحلیل رفتار و اثر متغیرها، ذخیره سازی فرمول‌ها و نمایش نمودارها.

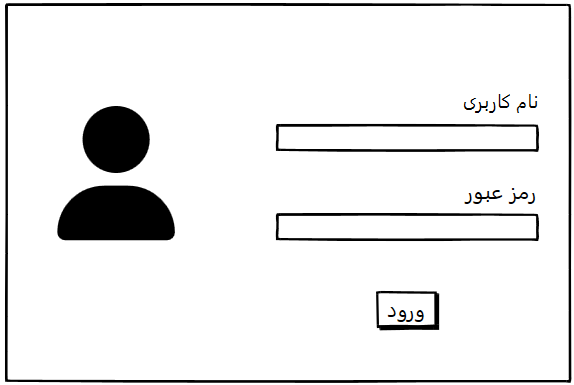
## 2-3. طراحی نمونه‌ی کاغذی[[18]](#footnote-18)

قدم بعدی برای ارتباط برقرار کردن مشتری با سامانه ما ساخت یک نمونه کاغذی برای نمایش عملکرد کلی و نسبت قرارگیری اجزای سامانه با یکدیگر است. به این صورت که با استفاده از نرم‌افزار بالزامیک[6] یک نمونه کاغذی ساخته شده تا چهارچوب کلی سامانه را بتوان به مشتری نشان داد و با توجه به بازخوردهای داده شده سامانه را ویرایش کرد و نیازمندی‌های دقیق و عملیاتی مشتری را استخراج کرد.



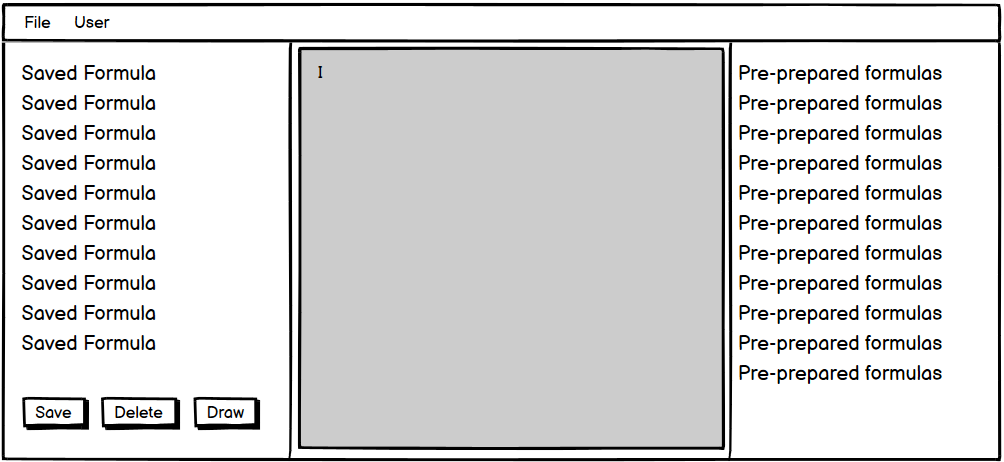
تصویر 3 نمونه ساخته شده توسط ابزار بالزامیک

نمای کلی از سامانه‌ی مورد نظر به صورت بالا خواهد بود. فرایند طراحی و ارائه آن یک هفته به طول انجامید و نسخه اولیه سامانه مطابق با تصویر بالا به مشتریان ارائه گشت. پس از جلسه دو ساعته با محوریت بررسی عملکرد سامانه، سامانه ما ویرایش شده و نسخه نهایی نمونه کاغذی مطابق با تصویر زیر است.



تصویر 4 صفحه‌ی ورود به سامانه

در بالا می‌توان صفحه‌ی ورود را مشاهده کرد. سپس به بررسی صفحه اصلی می‌پردازیم:



تصویر 5 تصویر نهایی نسخه کاغذی

همانطور که مشاهده می‌کنید منوها به تصویر اضافه شدند و ارتفاع صفحه کم شد.

## 3-3. معماری انتزاعی سامانه

به صورت کلی ساختار سامانه‌ی ما به صورت زیر است:



همانطور که مشاهده می‌کنید، ما یک ساختار برای احراز هویت لازم داریم که اطلاعات کاربران را در یک پایگاه داده ذخیره کند. یک ساز و کار برای دریافت فرمول‌های ورودی کاربران و نمایش آنها داریم. در مجموع ما دو واحد پردازشی (تبدیل ورودی کاربر به فرمول قابل استفاده و احراز هویت) و دو واحد ذخیره‌سازی (اطلاعات کاربران و فرمول‌های ذخیره شده) داریم.

## 4-3. مراحل انجام پروژه

### 1-4-3. انتخاب زبان برنامه نویسی

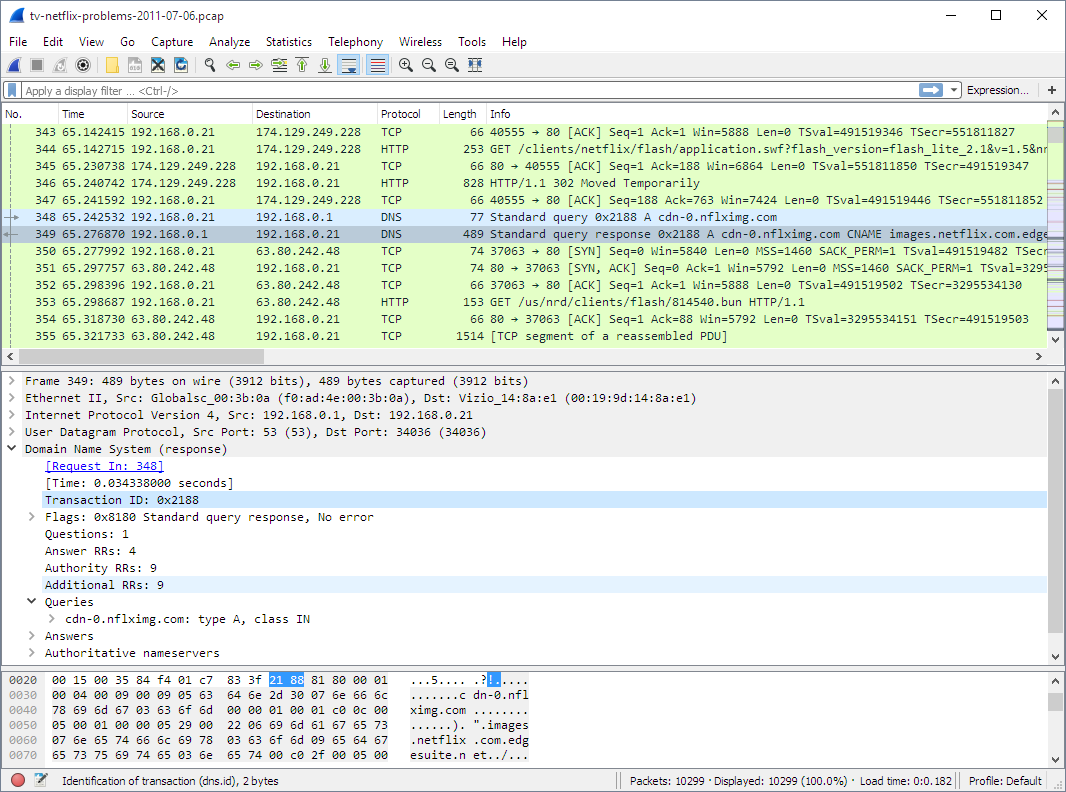
مرحله‌ی نخست انتخاب زبان برنامه‌نویسی مناسب برای انجام پروژه است. برای اینکار باید زبان‌هایی که دارای رابط کاربری هستند را مورد نقد و بررسی قرار دهیم تا ببینیم کدام یک برای انجام پروژه‌ی ما مناسب هستند.

#### 1-1-4-3. زبان C و C++

ساخت یک رابط کاربری با زبان‌های C و C++ یک کار چالش‌برانگیز است. این زبان‌ها به دلیل عملکرد بالا و دسترسی مستقیم به سخت‌افزار، برای برنامه‌هایی که نیاز به کارایی بالا دارند، مانند بازی‌ها، نرم‌افزارهای سیستم‌عامل، و برنامه‌های گرافیکی پیچیده، بسیار مناسب هستند.

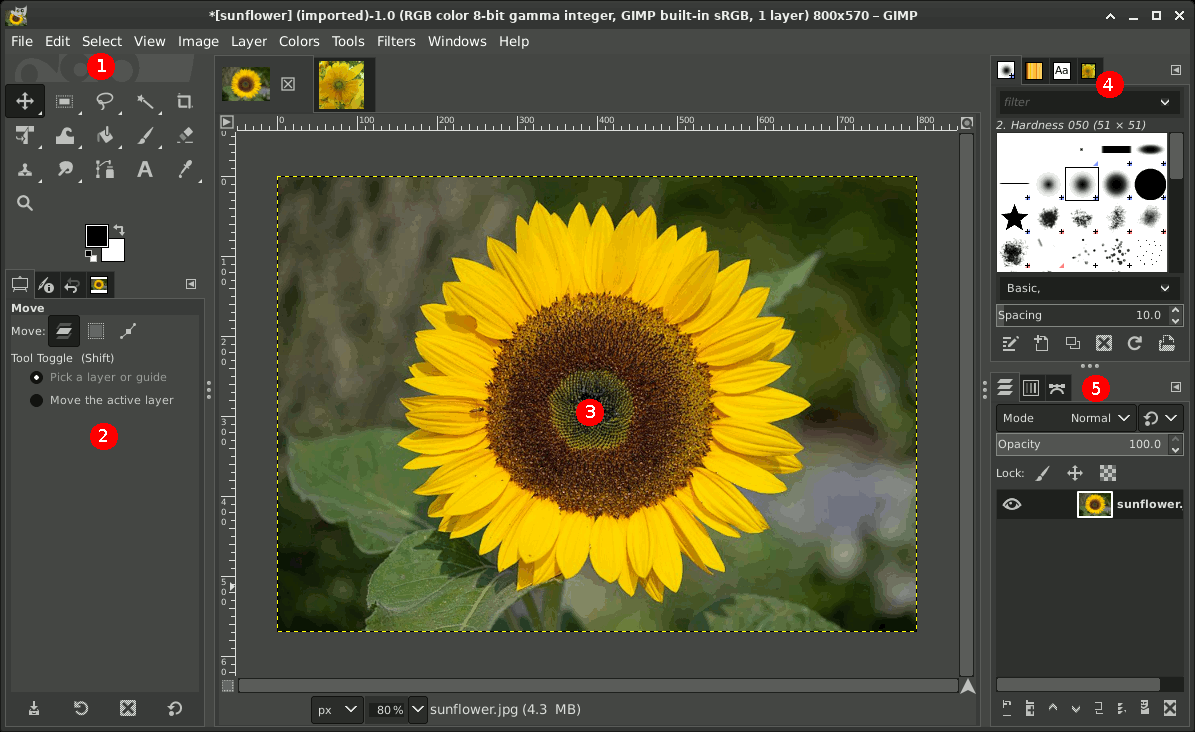
C و C++ به طور پیش‌فرض کتابخانه گرافیکی خاصی برای ساخت UI ندارند، بنابراین شما باید از یک کتابخانه گرافیکی خارجی استفاده کنید. چند کتابخانه رایج عبارتند از:

* Qt: یک کتابخانه قدرتمند و چند سکویی است که ابزارهای پیشرفته‌ای برای ساخت رابط کاربری فراهم می‌کند. از پروژه‌های معروفی که با این کتابخانه نوشته شده است می‌توان به VirtualBox، Telegram Desktop، Spotify و Wireshark اشاره کرد.



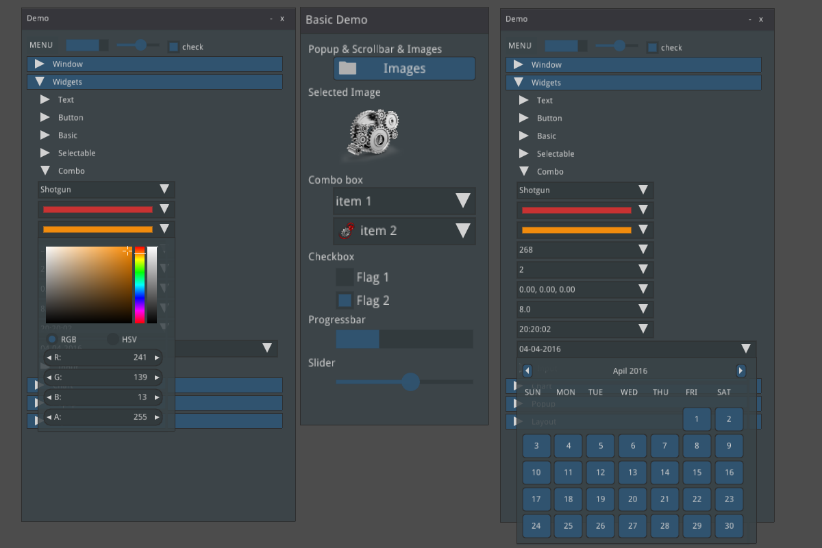
*تصویر 6 نمونه‌ای از فضای نرم‌افزار wireshark*

* GTK: بیشتر در محیط‌های لینوکس استفاده می‌شود و برای ساخت برنامه‌های گرافیکی کاربرد دارد. از این کتابخانه برای توسعه نرم‌افزارهای مختلفی مثل GIMP، Key sign و بسیاری از ابزارهای سیستم عامل لینوکس استفاده شده است.



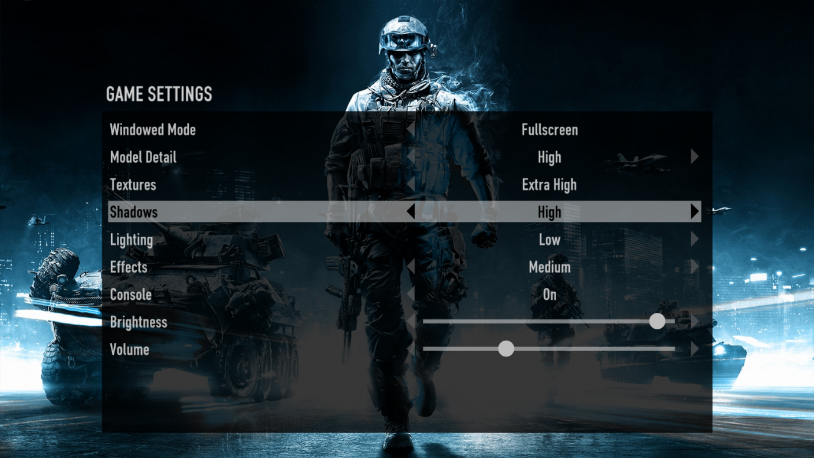
*تصویر 7 نمونه‌ای از فضای نرم‌افزار GIMP*

* wxWidgets: یک کتابخانه دیگر چند سکویی است که به شما امکان می‌دهد رابط‌های کاربری گرافیکی را در ویندوز، لینوکس و مک ایجاد کنید.
* SFML یا SDL: اگر به دنبال ساخت بازی یا برنامه‌های چندرسانه‌ای هستید، این کتابخانه‌ها مناسب‌تر هستند.
* Nuklear: این کتابخانه بدلیل آنکه از کتابخانه‌های رسمی نمی‌باشد، فاقد منابع غنی برای برخطایابی[[19]](#footnote-19) بوده و در صورت بروز هرگونه اشتباه یا مشکل در پیاده‌سازی رابط کاربری، زمان بسیار زیادی صرف درست کردن آن خواهد شد.



*تصویر 8 نمونه ای از فضای نرم‌افزار نوشته شده با کتابخانه‌ی Nuklear*

از مشکلات استفاده از Nuklear، می‌توان به عدم لایه‌ای بودن آن اشاره کرد و اینکه این کتابخانه به صورت اجزاء[[20]](#footnote-20) را قرار می‌دهد و امکان طراحی انتزاعی را از طراحان می‌گیرد که این کار توسعه را بسیار سخت و زمان‌بر می‌کند. از پروژه‌هایی که با این کتابخانه توسعه داده شده‌اند می‌توان به بازی Battlefield اشاره کرد.



*تصویر9 فضای بازی Battle Field*

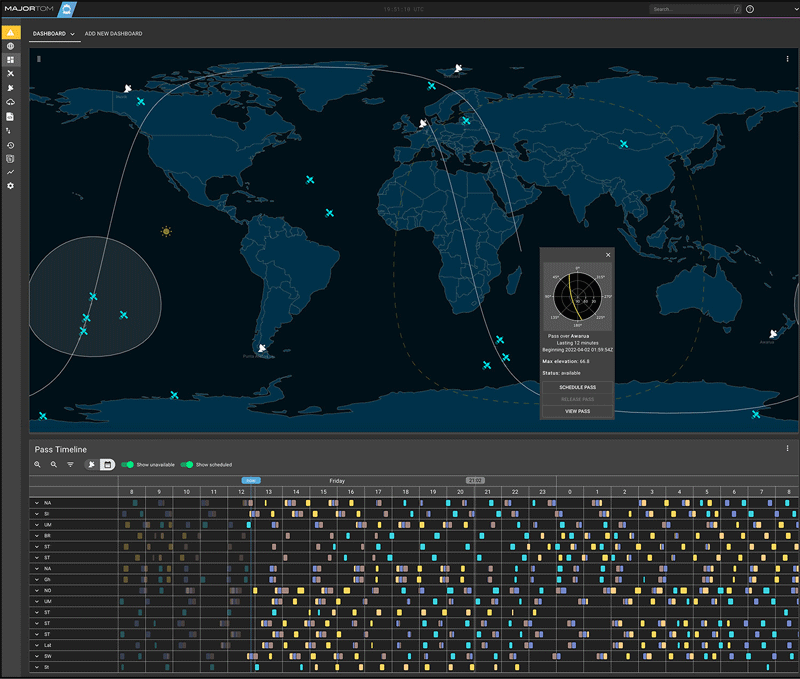
زبان C بدلایل زیر می‌تواند نماینده خوبی برای انجام کارهای محاسباتی باشد:

1. سرعت بالا
2. کنترل بر حافظه در پردازش‌های سنگین
3. استقلال از پلتفرم
4. کار با سخت‌افزار
5. برنامه‌نویسی موازی

#### 2-1-4-3. زبان جاوا

در زبان جاوا کتابخانه‌های زیادی برای طراحی رابط کاربری وجود دارد که می‌توان به موارد زیر اشاره کرد:

* JavaFX: یک فریم‌ورک مدرن برای توسعه برنامه‌های دسکتاپ با استفاده از زبان برنامه‌نویسی Java است. این فریم‌ورک توسط شرکت Oracle معرفی شد تا جایگزین تکنولوژی قدیمی‌تر Swing شود و به برنامه‌نویسان اجازه دهد رابط‌های کاربری گرافیکی (GUI) زیبا و مدرن ایجاد کنند. JavaFX امکان توسعه برنامه‌های گرافیکی با جلوه‌های بصری پیشرفته، انیمیشن‌ها و تعامل با رسانه‌ها را فراهم می‌کند اما بدلایلی که در آینده به آن‌ها خواهیم پرداخت (پردازش‌های محاسباتی) از زبان پایتون استفاده می‌کنیم.



*تصویر 10 نمونه ای نرم‌افزار پیاده شده با javafx*

* Swing: یکی از کتابخانه‌های Java برای ساخت رابط‌های کاربری گرافیکی است که بر اساس AWT توسعه داده شده است. Swing به‌عنوان بخشی از Java Foundation Classes (JFC) معرفی شد و به توسعه‌دهندگان این امکان را می‌دهد که برنامه‌های دسکتاپی با رابط‌های گرافیکی ایجاد کنند.
* JGoodies Forms: این کتابخانه به توسعه‌دهندگان کمک می‌کند تا چیدمان‌های (layouts) پیچیده و انعطاف‌پذیر برای برنامه‌های دسکتاپ بسازند، به‌ویژه زمانی که از فریم‌ورک‌های جاوا مانند Swing استفاده می‌کنند. در برنامه‌نویسی رابط کاربری، مدیریت چیدمان اجزا یکی از چالش‌های اصلی است، به‌خصوص وقتی که نیاز به یک طراحی منظم و واکنش‌گرا برای اندازه‌ها و ابعاد مختلف باشد. JGoodies Forms به‌منظور حل این مشکلات طراحی شده و راهکاری ساده‌تر و قدرتمندتر برای مدیریت چیدمان اجزا ارائه می‌دهد.

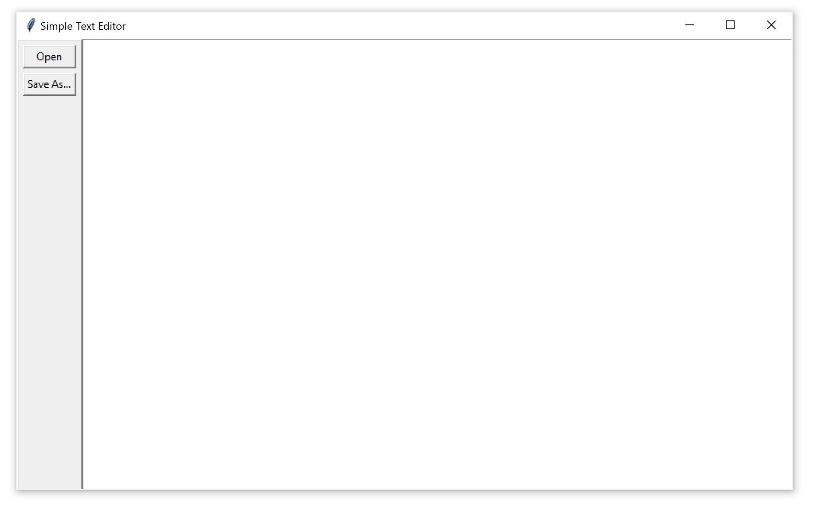
بعضی از کتابخانه‌ها بدلیل کم‌بود منابع (JGoodies Forms و JIDE Common Layer) و باقی بدلیل قدیمی شدن و عدم کاربرپسند بودن رابط‌های کاربری آن‌ها و هزینه زیاد بابت ترمیم این نقص از لیست پیشنهادات ما حذف می‌شوند.

#### 3-1-4-3. زبان پایتون

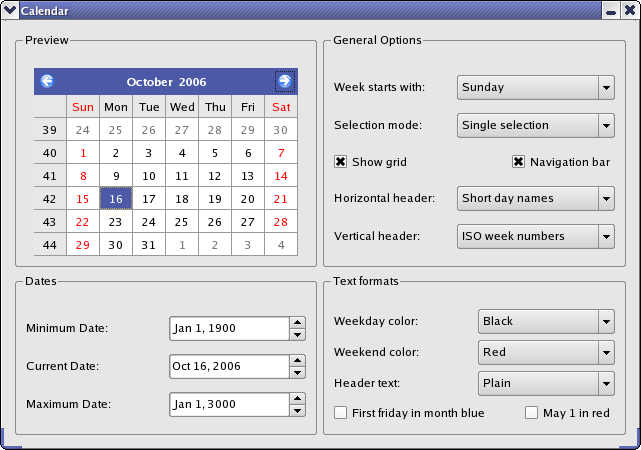
زبان برنامه نویسی پایتون، دارای پنج کتابخانه‌ی معروف برای طراحی رابط کاربری است:

* Tkinter: یک کتابخانه استاندارد برای ایجاد رابط‌های کاربری گرافیکی در پایتون است. این کتابخانه یک رابط به ابزار Tk GUI toolkit محسوب می‌شود که یکی از قدیمی‌ترین و پرکاربردترین ابزارهای طراحی رابط‌های گرافیکی است. Tkinter به‌طور پیش‌فرض با پایتون نصب می‌شود و برای ساخت برنامه‌های دسکتاپ ساده و کاربردی بسیار مفید است.
* PyQt: بر اساس فریم‌ورک Qt، یکی از معروف‌ترین ابزارهای ساخت برنامه‌های گرافیکی، توسعه داده شده است. PyQt امکانات گسترده‌ای برای طراحی رابط‌های کاربری مدرن، چندسکویی و پیچیده ارائه می‌دهد و همچنین شامل ابزارهای پیشرفته‌ای برای کار با تصاویر، گرافیک‌های دوبعدی و سه‌بعدی، سیستم‌های چندرسانه‌ای، و مدیریت رویدادهاست. این کتابخانه به توسعه‌دهندگان اجازه می‌دهد تا برنامه‌هایی با ظاهر و کارکرد حرفه‌ای در سیستم‌عامل‌های مختلف ایجاد کنند، اما به دلیل ویژگی‌های پیشرفته و انعطاف‌پذیری بالا، ممکن است یادگیری آن در مقایسه با ابزارهای ساده‌تر مانند Tkinter زمان‌برتر باشد.
* PySide: مانند PyQt بر اساس فریم‌ورک Qt توسعه داده شده است. PySide توسط شرکت Qt توسعه یافته و به‌طور رسمی پشتیبانی می‌شود. این کتابخانه به توسعه‌دهندگان امکان می‌دهد تا برنامه‌های گرافیکی چندسکویی و مدرن بسازند که در سیستم‌عامل‌های مختلف مانند ویندوز، مک و لینوکس قابل اجرا هستند. تفاوت اصلی PySide با PyQt در مجوز استفاده آن‌هاست؛ PySide تحت مجوز LGPL منتشر می‌شود که استفاده از آن در پروژه‌های تجاری را آسان‌تر می‌کند. PySide و PyQt از نظر کارکرد بسیار مشابه هستند و هر دو قابلیت‌های غنی Qt را در پایتون ارائه می‌دهند.
* Kivy:
* wxPython:

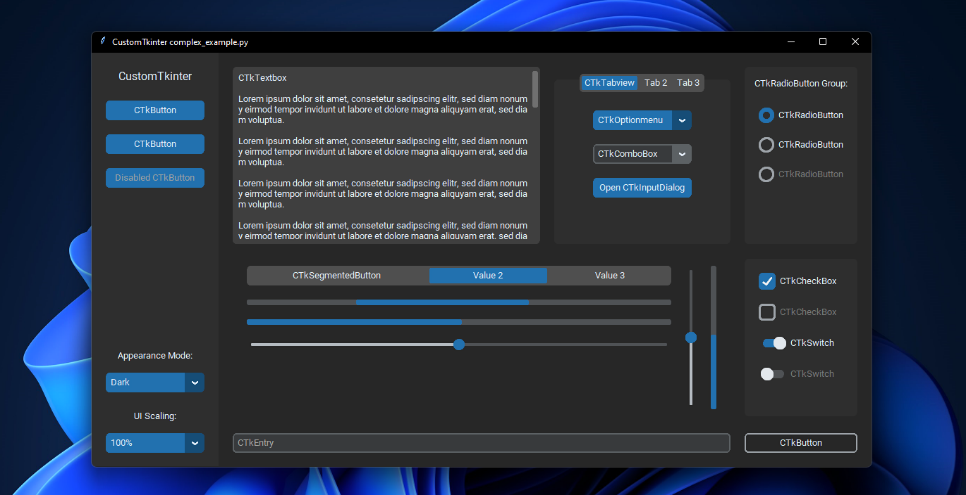
یکی از کتابخانه‌های غیر معروف زبان پایتون، کتابخانه‌ی Custom-Tkinter است که از آن می‌توان برای طراحی رابط‌های کاربری حرفه‌ای استفاده کرد. در ادامه به معرفی کتابخانه‌های مذکور خواهیم پرداخت.



*تصویر 11 نمونه ساخته شده به وسیله Tkinter*

ا

*تصویر 12 نمونه ساخته شده به وسیله Qt*



*تصویر 13 نمونه ساخته شده توسط کتابخانه Custom-Tkinter*

### 2-4-3. زمانبندی چرخه‌ها

برای زمان‌بندی پروژه، هر چرخه به بازه‌های دوهفته‌ای تقسیم‌بندی شده و

## نتیجه‌گیری

انتخاب زبان برنامه‌نویسی به شدت بستگی به کاربرد و استفاده از آن دارد. با توجه به اینکه در این پروژه تمرکز اصلی به روی سادگی و قابل فهم بودن و در عین حال ترسیم ساده است، زبان برنامه‌نویسی پایتون برای این کار انتخاب شده شد.

|  |  |  |  |
| --- | --- | --- | --- |
| زبان برنامه نویسی | رفع اشکال و اصلاح | قدرت محاسباتی | زیبایی |
| C/C++ | سخت | متوسط | بله |
| Java | ساده | پایین | بله |
| Python | ساده | بسیار بالا | بله |

# **فصل چهارم**

# 4. طراحی و پیاده‌سازی

## 1-4 رابط کاربری

همانظور که در قسمت قبل گفته شد، برای پیاده‌سازی رابط کاربری از کتابخانه‌ی custom-tkinter استفاده شده است که از نظر تجربه کاربری، عملکرد بهتری دارد. نرم‌افزار ما دارای سه صفحه‌ی اصلی است: صفحه‌ی ورود، صفحه‌ی نوشتن فرمول‌ها و صفحه تنظیمات.

صفحه‌ی تنظیمات باید متناسب با سه شخصیت مذکور در ابتدای گزارش شخصی‌سازی شده و تنظیمات نمایش داده شده به هر کدام متفاوت باشد. صفحه‌ی نوشتن فرمول‌ها و صفحه‌ی ورود برای همه افراد یکسان است؛ به استثنای دانشجویان که امکان ذخیره‌سازی فرمول‌ها را ندارند.

## 2-4 پیاده‌سازی رابط کاربری

.

.

## 3-4 ساختمان داده‌ها

.

.

## نتیجه‌گیری

.

.

# فصل پنجم

# 5. تحلیل و بررسی میزان بهره وری

## 1-5. ذخیره‌سازی اطلاعات

.

.

## 2-5. بررسی آزمون‌های واحد

.

.

## نتیجه‌گیری

.

.

# فصل ششم

# نتیجه‌گیری و پیشنهادات

## نتیجه‌گیری

.

.

## پیشنهادات

.

.

# مراجع

[1]. Tufte, Edward .R. *The Visual Display of Quantitative Information.* 1983.

[2]. Friendly, Michael. "A Brief History of Data Visualization." (2008).

[3]. Harvey Deitel, Paul J. Deitel, Jonathan P. Liperi, Ben Wiedermann. Python How to Program (Parts a & B). PAP/CDR ed. Vol. 1: Pearson, February 14, 2002, 2002.

[4]. Roger S. Pressman, Bruce Maxim. Software Engineering: A Practitioner's Approach. 1 vols. Vol. 1: McGraw Hill, January 23, 2014, 2019. Engineering.

[5]. Rubin, Kenneth. Essential Scrum: A Practical Guide to the Most Popular Agile Process (Addison-Wesley Signature Series (Cohn)). 1 st ed. Vol. 1: Addison-Wesley Professional, July 26, 2012.

[6]. https://balsamiq.com

Chaudhuri, Surajit, Umeshwar Dayal, and Vivek Narasayya. "An Overview of Business Intelligence Technology." *Communications of the ACM* 54, no. 8 (2011): 88-98.

Tufte, Edward .R. *The Visual Display of Quantitative Information.* 1983.

1. Agile [↑](#footnote-ref-1)
2. Sprint [↑](#footnote-ref-2)
3. William Henry Playfair [↑](#footnote-ref-3)
4. Agile [↑](#footnote-ref-4)
5. Sprint [↑](#footnote-ref-5)
6. Demo [↑](#footnote-ref-6)
7. Live Demo [↑](#footnote-ref-7)
8. Video Demo [↑](#footnote-ref-8)
9. Interactive Demo [↑](#footnote-ref-9)
10. Slide-Based Demo [↑](#footnote-ref-10)
11. Remote Demo [↑](#footnote-ref-11)
12. Waterfall [↑](#footnote-ref-12)
13. Scrum [↑](#footnote-ref-13)
14. Agile Manifesto [↑](#footnote-ref-14)
15. Sprint Backlog [↑](#footnote-ref-15)
16. Increment [↑](#footnote-ref-16)
17. Use Case [↑](#footnote-ref-17)
18. Sketch [↑](#footnote-ref-18)
19. Debugging [↑](#footnote-ref-19)
20. Modular [↑](#footnote-ref-20)