دانشکدهٔ فنی و مهندسی

گروه مهندسی کامپیوتر و فناوری اطلاعات

کارشناسی ارشد مهندسی کامپیوتر نرم‌افزار

گزارش سمینار

طراحی نرم‌افزار خدمات دانشجویی

نگارش:

محمد امانعلیخانی

استاد راهنما:

دکتر سید علی رضوی ابراهیمی

خردادماه ۱۴۰۰


بسم الله الرحمن الرحيم

چکیده

سیستم‌های کامپیوتری در مدت زمانی که وارد جامعه ما شده‌اند جای خود را در میان افراد جامعه باز کرده‌اند. همچنین جامعه نیز نیاز به این سیستم‌ها را احساس کرده است تا بتواند با این سیستم‌ها کارها را سریع‌تر انجام دهد. سیستم‌های کامپیوتری نیز نرم‌افزارهای پیشرفته‌تری را طلب می‌کنند،از این رو بر ماست که با ساخت نرم افزارهای لازم این نیازها را برطرف کنیم. سیستم خدمات دانشجویی نیز می‌تواند یکی از این نرم‌افزارها باشد تا بتواند قسمتی از این نیازها را برطرف کند. از طرفی برای تولید این نرم‌افزارها نیاز به دانستن زبان برنامه نویسی می‌باشد که زبان NodeJS نیز یکی از این زبان‌ها می‌باشد که با فرا گرفتن مهارت‌های لازم در این زبان می‌توان به سادگی نرم‌افزارهای مورد نیاز را تولید کرد. سیستم خدمات دانشجویی برای تعریف پروژه(کارفرما) و انجام پروژه(فریلنسر) می‌باشد که دارای امکاناتی از قبیل درج پروژه، پذیرش دانشجویان، جستجو در میان دانشجویان و جستجو در میان پروژه‌ها و ویرایش کردن و تصحیح اطلاعات و ... می‌باشد. در گذشته برای کسب درآمد نیاز به کار در بیرون از منزل مرسوم بوده و اشتغال در منزل تعریف به خصوصی نداشت. اما اخیرا در چند سال گذشته با افزایش چشمگیر فضای اینترنت و در طبع آن کسب‌وکارهای اینترنتی، مشاغل خانوادگی از جمله فریلنسینگ مطرح شد. شغل‌های فریلنسری نیاز به معرفی حضوری ندارند و کسانی که مهارت کافی را در زمینه فعالیت خود داشته باشند بدون شک خواهند توانست در این زمینه موفق شوند و درآمد خوبی را کسب کنند.

کلمات کلیدی: فریلنسر، کارفرما، سایت خدماتی، مشتری، آزادکار، پروژه‌محور

فهرست مطالب

فصل ۱: مقدمه

مقدمه ۲
۱.۱ تعریف و بیان کار ۲
۲.۱ سابقه و ضرورت ۲
۳.۱ هدف‌ها ۴
۴.۱ کاربردها ۴
۵.۱ مراحل کار ۴
۶.۱ روش کار ۴
۷.۱ ساختار گزارش ۴

فصل ۲: مفاهیم و دانش فنی

مقدمه ۷
۱.۲ BackEnd ۷
۱.۱.۲ JavaScript ۷
۲.۱.۲ NodeJS ۹
۳.۱.۲ NoSQL ۱۱
۴.۱.۲ MongoDB ۱۲
۲.۲ FrontEnd ۱۳
۱.۲.۲ Html ۱۴
۲.۲.۲ Css ۱۵
۳.۲.۲ Scss ۱۶
۴.۲.۲ Figma ۱۷
۳.۲ الگوی معماری MVC ۱۸
۱.۳.۲ MVC چیست ؟ ۱۸
۲.۳.۲ تاریخچه MVC ۱۸
۳.۳.۲ ویژگی های MVC ۱۸
۴.۳.۲ اجزای MVC ۱۹
۵.۳.۲ مزایای MVC ۱۹
۶.۳.۲ معایب MVC ۲۰

فصل ۳: تجزیه و تحلیل نرم‌افزار

مقدمه ۲۳
۱.۳ ثبت نام ۲۳
۲.۳ ورود ۲۸
۳.۳ لیست پروژه‌ها ۳۳
۴.۳ لیست فریلنسرها ۳۶
۵.۳ داشبورد کاربر ۳۹
۶.۳ داشبورد کارفرما ۴۲
۱.۶.۳ ایجاد پروژه ۴۳
۲.۶.۳ ویرایش پروژه ۴۷
۳.۶.۳ حذف پروژه ۵۰
۴.۶.۳ پرداخت هزینه پروژه ۵۳
۵.۶.۳ واگذاری پروژه ۵۶
۷.۳ داشبورد فریلنسر ۵۹
۱.۷.۳ ایجاد رزومه ۵۹
۲.۷.۳ ویرایش رزومه ۶۳
۳.۷.۳ حذف رزومه ۶۵
۴.۷.۳ درخواست پروژه ۶۸
۵.۷.۳ دریافت هزینه پروژه ۷۱
۸.۳ داشبورد مدیریت ۷۵
۱.۸.۳ مدیریت مالی ۷۶
۲.۸.۳ مدیریت اختلافات ۷۶
۳.۸.۳ مدیریت کاربران ۷۷

فصل ۴: طراحی و پیاده‌سازی نرم‌افزار

مقدمه ۷۹
۱.۴ پوشه تنظیمات (config) ۷۹
۲.۴ پوشه کنترل (controllers) ۸۰
۱.۲.۴ فایل middleware ۸۰
۲.۲.۴ فایل auth ۸۲
۳.۲.۴ فایل employer ۸۵
۴.۲.۴ فایل frelanser ۹۰
۳.۴ پوشه مدل (models) ۹۴
۱.۳.۴ فایل user ۹۴
۲.۳.۴ فایل employer ۹۵
۳.۳.۴ فایل frelanser ۹۵
۴.۴ پوشه مسیر (routes) ۹۶

۱.۴.۴ فایل index . . . ۹۶
۲.۴.۴ فایل home . . . ۹۶
۳.۴.۴ فایل auth . . . ۹۶
۴.۴.۴ فایل dashboard . . . ۹۶
۵.۴.۴ فایل employer . . . ۹۷
۶.۴.۴ فایل frelanser . . . ۹۷
۵.۴ پوشه نما (views) . . . ۹۷
۶.۴ پوشه عمومی (public) . . . ۹۸
۷.۴ فایل اصلی/اجرا (app.js) . . . ۹۸
۸.۴ پوشه (husky.) . . . ۹۸
۹.۴ پوشه (vscode.) . . . ۹۸
۱۰.۴ پوشه مستندات (docs) . . . ۹۸
۱۱.۴ پوشه گیت‌هاب (github.) . . . ۹۹
۱۲.۴ فایل Environment (env.) . . . ۹۹
۱۳.۴ فایل تاریخچه تغییرات (CHANGELOG) . . . ۱۰۰
۱۴.۴ فایل Travis CI (travis.) . . . ۱۰۰
۱۵.۴ فایل Editor Config (editorconfig.) . . . ۱۰۰
۱۶.۴ فایل ESLint (eslintrc.) . . . ۱۰۰
۱۷.۴ فایل Prettier (prettierrc.) . . . ۱۰۱
۱۸.۴ ساختار پایگاه‌داده . . . ۱۰۱
۱.۱۸.۴ پایگاه‌داده کاربر . . . ۱۰۱
۲.۱۸.۴ پایگاه‌داده کارفرما . . . ۱۰۲
۳.۱۸.۴ پایگاه‌داده فریلنسر . . . ۱۰۳
۴.۱۸.۴ پایگاه‌داده مدیریت . . . ۱۰۳

فصل ۵: جمع‌بندی و پیشنهادات

مقدمه . . . ۱۰۵

واژه‌نامه

فارسی به انگلیسی . . . ۱۰۷
انگلیسی به فارسی . . . ۱۰۸

فهرست تصاویر

۱.۳ دیاگرام UC ساختار کلی ۲۳
۲.۳ دیاگرام فعالیت ثبت‌نام ۲۵
۳.۳ دیاگرام حالت ماشین ثبت‌نام ۲۶
۴.۳ دیاگرام توالی ثبت‌نام ۲۷
۵.۳ دیاگرام همکار ثبت‌نام ۲۸
۶.۳ دیاگرام فعالیت ورود ۳۰
۷.۳ دیاگرام حالت ماشین ورود ۳۱
۸.۳ دیاگرام توالی ورود ۳۲
۹.۳ دیاگرام همکار ورود ۳۳
۱۰.۳ دیاگرام فعالیت لیست پروژه‌ها ۳۴
۱۱.۳ دیاگرام حالت ماشین پروژه‌ها ۳۴
۱۲.۳ دیاگرام توالی پروژه‌ها ۳۵
۱۳.۳ دیاگرام همکار پروژه‌ها ۳۶
۱۴.۳ دیاگرام فعالیت فریلنسرها ۳۷
۱۵.۳ دیاگرام حالت ماشین فریلنسرها ۳۷
۱۶.۳ دیاگرام توالی فریلنسرها ۳۸
۱۷.۳ دیاگرام همکار فریلنسرها ۳۹
۱۸.۳ دیاگرام UC داشبورد کاربر ۴۰
۱۹.۳ دیاگرام فعالیت داشبورد کاربر ۴۰
۲۰.۳ دیاگرام حالت ماشین داشبورد کاربر ۴۱
۲۱.۳ دیاگرام توالی داشبورد کاربر ۴۱
۲۲.۳ دیاگرام همکار داشبورد کاربر ۴۲
۲۳.۳ دیاگرام UC داشبورد کارفرما ۴۳
۲۴.۳ دیاگرام فعالیت ایجاد پروژه ۴۴
۲۵.۳ دیاگرام حالت ماشین ایجاد پروژه ۴۵
۲۶.۳ دیاگرام توالی ایجاد پروژه ۴۶
۲۷.۳ دیاگرام همکار ایجاد پروژه ۴۷
۲۸.۳ دیاگرام فعالیت ویرایش پروژه ۴۸

۲۹.۳ دیاگرام حالت ماشین ویرایش پروژه ۴۹
۳۰.۳ دیاگرام توالی ویرایش پروژه ۴۹
۳۱.۳ دیاگرام همکار ویرایش پروژه ۵۰
۳۲.۳ دیاگرام فعالیت حذف پروژه ۵۱
۳۳.۳ دیاگرام حالت ماشین حذف پروژه ۵۲
۳۴.۳ دیاگرام توالی حذف پروژه ۵۲
۳۵.۳ دیاگرام همکار حذف پروژه ۵۳
۳۶.۳ دیاگرام فعالیت پرداخت هزینه ۵۵
۳۷.۳ دیاگرام حالت ماشین پرداخت هزینه ۵۵
۳۸.۳ دیاگرام توالی پرداخت هزینه ۵۵
۳۹.۳ دیاگرام همکار پرداخت هزینه ۵۶
۴۰.۳ دیاگرام فعالیت واگذاری پروژه ۵۷
۴۱.۳ دیاگرام حالت ماشین واگذاری پروژه ۵۷
۴۲.۳ دیاگرام توالی واگذاری پروژه ۵۸
۴۳.۳ دیاگرام همکار واگذاری پروژه ۵۸
۴۴.۳ دیاگرام UC داشبورد فریلنسر ۵۹
۴۵.۳ دیاگرام فعالیت ایجاد رزومه ۶۱
۴۶.۳ دیاگرام حالت ماشین ایجاد رزومه ۶۱
۴۷.۳ دیاگرام توالی ایجاد رزومه ۶۲
۴۸.۳ دیاگرام همکار ایجاد رزومه ۶۲
۴۹.۳ دیاگرام فعالیت ویرایش رزومه ۶۴
۵۰.۳ دیاگرام حالت ماشین ویرایش رزومه ۶۴
۵۱.۳ دیاگرام توالی ویرایش رزومه ۶۵
۵۲.۳ دیاگرام همکار ویرایش رزومه ۶۵
۵۳.۳ دیاگرام فعالیت حذف رزومه ۶۶
۵۴.۳ دیاگرام حالت ماشین حذف رزومه ۶۷
۵۵.۳ دیاگرام توالی حذف رزومه ۶۷
۵۶.۳ دیاگرام همکار حذف رزومه ۶۸
۵۷.۳ دیاگرام فعالیت درخواست پروژه ۶۹
۵۸.۳ دیاگرام حالت ماشین درخواست پروژه ۷۰
۵۹.۳ دیاگرام توالی درخواست پروژه ۷۰
۶۰.۳ دیاگرام همکار درخواست پروژه ۷۱
۶۱.۳ دیاگرام فعالیت دریافت وجه ۷۲
۶۲.۳ دیاگرام حالت ماشین دریافت وجه ۷۳
۶۳.۳ دیاگرام توالی دریافت وجه ۷۴
۶۴.۳ دیاگرام همکار دریافت وجه ۷۵

۶۵.۳ دیاگرام UC داشبورد مدیریت ۷۶

۱.۴ ساختار کلی ۷۹
۲.۴ ساختار پوشه تنظیمات ۷۹
۳.۴ ساختار پوشه کنترل ۸۰
۴.۴ ساختار پوشه مدل ۹۴
۵.۴ ساختار پوشه مسیر ۹۶
۶.۴ ساختار پوشه نما ۹۷
۷.۴ ساختار پوشه عمومی ۹۸
۸.۴ ساختار پوشه مستندات ۹۹
۹.۴ ساختار قرارگیری گیت‌هاب ۹۹
۱۰.۴ ساختار پایگاه داده ۱۰۱
۱۱.۴ ساختار پایگاه داده کاربر ۱۰۲
۱۲.۴ ساختار پایگاه داده کارفرما ۱۰۳
۱۳.۴ ساختار پایگاه داده فریلنسر ۱۰۳
۱۴.۴ ساختار پایگاه داده مدیریت ۱۰۳

فهرست علائم اختصاری

HTML HyperText Markup Language

CSS Cascading Style Sheets

Sass Syntactically Awesome Style Sheets

JS JavaScript . . . . . جاوا اسکریپت

JSX JavaScript XML

JSON JSON

NoSQL Not Only SQL

SPA Single-Page Application . . . . . اپلیکیشن تک صفحه‌ای

RTA Real-Time Application . . . . . اپلیکیشن بی‌درنگ

TDD Test Driven Development . . . . . توسعه آزمون‌محور

ENV Environment Variables . . . . . متغیرهای محیطی

CI Continuous Integration . . . . . یکپارچه‌سازی مداوم / ادغام مداوم

CD Continuous Delivery / Continuous Deployment . . . . . تحویل مداوم / استقرار مداوم

MVC Model View Controller . . . . . مدل نما کنترل

CONFIG Configuration . . . . . تنظیمات

DOCS Documents . . . . . مستندات

فصل ۱

مقدمه

مقدمه

امروزه با پیشرفت تکنولوژی و افزایش ضریب استفاده از اینترنت، شکل و مفهوم کار دستخوش تغییرات زیادی شده است. این تغییرات فرصت‌های فراوانی را در اختیار کارآفرینان، متخصصین و افراد علاقه‌مند به تکنولوژی‌های جدید می‌گذارد. فریلنسینگ یا دورکاری، شیوه‌ای از کار است که در آن متخصصین، بدون نیاز به حضور دائمی در دفتر کار یا تعهد بلندمدت به یک شرکت، به‌صورت پروژه‌ای، خدمات حرفه‌ای خود را به کسب‌وکارها ارائه می‌دهند. همگام شدن با تغییرات و شیوه‌های جدید، رمز موفقیت هر فرد یا سازمان موفقی است. به‌منظور تسهیل ارائه خدمات و افزایش رقابت‌پذیری و همچنین فراهم آوردن فرصت کاری برابر برای متخصصین، اقدام به طراحی پلتفرم برون‌سپاری پروژه‌های کوچک و بزرگ نموده است. متخصصین فریلنسر یا دورکار، با مهارت‌هایی مانند برنامه‌نویسی، طراحی سایت، طراحی گرافیک و دیزاین، تولید محتوا، ترجمه، سئو و سایر مهارت‌ها قادرند بدون محدودیت‌های معمول مکانی یا زمانی، با ارائه خدمات خود به صورت آنلاین و اینترنتی کسب درآمد نمایند. همچنین از اهداف ما فراهم آوردن امکان کسب درآمد از طریق دورکاری برای معلولین و افراد کم‌توان بوده است. کارآفرینان و صاحبان کسب‌وکار با امکان دسترسی به بانک اطلاعاتی بزرگی از متخصصین از تمامی نقاط می‌توانند خدمات حرفه‌ای موردنیاز خود را باکیفیت بهتر و قیمت مناسب‌تری تحویل بگیرند و بهره‌وری و رقابت‌پذیری خود را افزایش دهند. این خدمات می‌تواند شامل طراحی وب‌سایت، برنامه‌نویسی، ساخت اپلیکیشن موبایل، طراحی لوگو و کاتالوگ، تهیه محتوای نوشتاری و تصویری، ساخت تیزر تبلیغاتی، تایپ، ترجمه متون و سایر خدمات حرفه‌ای باشد. هدف اصلی فراهم آوردن امکان دسترسی به بازار کار و اشتغال برای متخصصین در گوشه و کنار کشور حتی در شهرها و روستاهای کوچک و دورافتاده است. ما معتقدیم مهارت و دانش در همه‌جا وجود دارد و برابر بودن فرصت‌های کاری برای افراد باعث رونق و شکوفایی بیشتر اقتصادی می‌شود. معتقدیم فعالیت ما ضمن کمک به افزایش اشتغال‌زایی و ایجاد فرصت‌های کاری برابر، بستری تعاملی و دسترسی سریع و حرفه‌ای برای کسب‌وکارها، صاحبان ایده و کارآفرینان برای رفع نیازهای حرفه‌ای و ارتقا کسب‌وکار ایجاد می‌کند.

۱.۱ تعریف و بیان کار

به‌عنوان یک سایت فریلنسینگ امکان کسب درآمد از طریق دورکاری رو برای فریلنسرها فراهم کرده. همچنین برون‌سپاری پروژه‌های طراحی وب و برنامه‌نویسی، تولید محتوا، طراحی گرافیک، تایپ، ترجمه و بسیاری مهارت‌های دیگر هم از این طریق امکان‌پذیر است.

در این پروژه موارد زیر از دانشجو درخواست می‌شد:

۱. سیستم توانایی تعریف و پرداخت هزینه انجام پروژه را دارا باشد.

۲. این سیستم دارای انواع گزارش گیری از فریلنسرها و کارفرماها باشد.

۳. سیستم توانایی تعامل کارفرما و فریلنسر را دارا باشد.

۲.۱ سابقه و ضرورت

سابقه فریلنسری به جک نیلز برمی‌گردد. Nilles jack برای اولین بار ایده ارتباط از راه دور را مطرح کرد. در آن روزها یعنی دهه ۱۹۷۰ مشکل حمل‌ونقل در شهرها وجود داشت و کاملاً آزاردهنده بود که افراد ساکن مناطق دور برای کارهای روزمره

خود به شهرهای اصلی سفر کنند. حتی دولت نگران این موضوع بود و به دنبال راه‌حلی برای آن. همیشه گفته می‌شود که نیاز مادر اختراع است. با اختراع تلفن توسط گراهام بل در سال ۱۸۷۶ تمام دنیا شگفت‌زده شد. در دهه ۱۹۷۰ تلفن برای کار در دفاتر ایالات‌متحده امریکا مورداستفاده قرار گرفت و این همان چیزی بود که Nilles jack آن را مورد هدف قرارداد و به این فکر افتاد که از تلفن برای حل مشکل استفاده کند و درنهایت مفهوم کار در منزل با تلفن را معرفی کرد. آن زمان اینترنت هنوز دنیا را شگفت‌زده نکرده بود و بسیاری از این ایده کار کردن دلگیر شدند. نگرانی‌های زیادی در مورد کار در منزل وجود داشت و باید ثابت می‌شد که این نوع کار کردن مؤثر است. بنابراین او در آزمایشگاه دانشگاه کالیفرنیا شروع به انجام آزمایشاتی کرد و توانست نتایج خوبی را کسب کند. این‌گونه بود که وی توانست کمک‌های مالی خود را از موسسه ملی علوم دریافت کند و حتی دولت هم از این ایده حمایت کرد، چون‌که تنها راه‌حل برای حل مشکل حمل‌ونقل بود و می‌توانست مصرف سوخت را کاهش دهد و میزان افزایش آلودگی در آمریکا را کاهش دهد. Schiff Frank یکی دیگر از افرادی بود که از کار در خانه حمایت کرد. وی عمدتا نگران افزایش مصرف بنزین و سایر سوخت ها بود. او به پیشنهادات Nilles jack کمک کرد و به عنوان رئیس کمیته توسعه اقتصادی از ایده کار از راه دور دفاع کرد. او در مقاله ای با عنوان "کار در خانه می تواند بنزین را نجات دهد" به توصیفِ زیبایی مزایای کار از راه دور پرداخت. وی بیشتر این کار را با نام " محل کار انعطاف پذیر" مطرح می کرد. حتی در آن روز ها وی توانست مفهوم استفاده از اینترنت را به صورت بسیار تحریک آمیزی مطرح کند. در نتیجه این مقاله در ایالات متحده محبوبیت زیادی به دست آورد و ایده فریلنسری موجب شد که همه به بررسی مزایا و معایب فریلنسری بپردازند. Gordon Gil نام دیگری است که در اینجا باید به آن اشاره کنیم، وی کمک زیادی به انتشار ارتباط از راه دور کرد و در کنار جک نیلز دیدگاه های علمی در مورد مزایای دورکاری یا فریلنسری مطرح کرد. با گذشت زمان استفاده از اینترنت روز به روز بیشتر رواج یافت. در اواخر قرن بیست و بیست و یکم افراد فریلنسر برای انجام کاری خاص با یکدیگر ارتباط برقرار می کنند و وقتی که کار انجام می گرفت اتصالات منحل می شد و بدین ترتیب آنها دوباره آزاد می شدند. از آنجا که این افراد از طریق اینترنت به یکدیگر وصل می شدند اصلاح جدید e-lancers متولد شد. در سال های اولیه قرن بیست و یکم، کل حوزه اینترنتی با سرعت غیر قابل تصوری رونق پیدا کرد و اکنون تقریبا همه به اینترنت دسترسی دارند. اولین بار در سال ۱۹۹۹ و امروز در حدود بیش از ۱۰۰۰ پرتابل فریلنسری در اینترنت تاسیس شده است. با افزایش تعداد چنین سایت ها و شرکت هایی رقابت در بازار به صورت تصاعدی در حال افزایش است و این برای مشتری و فریلنسر یک مزیت محسوب می شود. فریلنسرها می توانند از این فرصت استفاده کنند تا بازارهایی که دارای درآمد خوبی هستند را پیدا کنند و مشتریان هم می توانند به دنبال پلت فرم هایی باشند که کمترین هزینه را برای آنها دارند. مشتری ها به طور کلی به فریلنسرهای با تجربه اعتماد دارند و ممکن است پروژه های مختلف خود را به آنها بسپارند. بنابراین بسیار مهم است که در زمینه های خاص تخصص کافی را بدست آورید. بعضی از مشتریان حاضرند برای یک کار با کیفیت بالا هزینه های خوبی پرداخت کنند. بنابراین برای اینکه این اتفاق بیافتد، برخورد خوب یکی از موضوعات ضروری است که باید در نظر داشته باشید. یکی دیگر از عواملی که می تواند تا حد زیادی روی درآمد یک فریلنسر تاثیر بگذارد زمینه ای است که وی برای کار انتخاب می کند. برای مثال یک برنامه نویس نرم افزار درآمد بسیار بیشتری نسبت به یک محتوا نویس کسب می کند. دلیل این امر هم این است که تعداد محتوانویسان بسیار زیاد است و تعداد برنامه نویسان نرم افزاری کم. عرضه کمتر باعث افزایش تقاضا و در نتیجه افزایش قیمت ها می شود. امیدواریم این تاریخچه فریلنسری مختصر نظر شما را به خود جلب کرده باشد.

- سایت‌های فریلنسری به کارفرمایان این امکان را می‌دهند که از میان هزاران فرد متخصص در یک حوزه خاص بتوانند پروژه خود را با قیمتی مشخص به فرد مورد نظر تحویل دهند
- کارفرما نیاز ندارد تا برای انجام یک یا چند پروژه، کارمند استخدام کند و می‌تواند با برونسپاری پروژه، کار را به فریلنسرها بسپارد

- به دلیل محیط رقابتی‌ای که در سایت‌های فریلنسری وجود دارد، کارفرمایان می‌توانند با قیمتی مناسب‌تر پروژه‌های خود را تحویل دهند. (البته در صورت کاهش بیش‌از حد قیمت افراد مجرب به انجام پروژه شما ترغیب نخواهند شد)
- کارفرمایان می‌توانند از میان فریلنسرها، کارمندان مناسبی را انتخاب کنند و آنها را از مزایای آن شغل مطلع کنند.

۳.۱ هدف‌ها

به راحتی بتوان برای پروژه‌های خود، نیروی متخصص پیدا کرد. پلتفرم فریلنسینگ و دورکاری اینترنتی، جهت استخدام فریلنسرهای حرفه‌ای در ایران است. از جمله پروژه‌هایی که می‌توانید آن‌ها را برون سپاری کنید و یا انجام دهید، می‌توان به: برنامه نویسی، ساخت اپلیکیشن و نرم افزار، طراحی سایت، تولید محتوای سایت و شبکه‌های اجتماعی، تایپ و ترجمه، عکاسی، طراحی (لوگو، بنر، پوستر، چهره و...)، نگارش مقاله و پروپوزال، بازاریابی، دیجیتال مارکتینگ، سئو SEO (بهینه‌سازی موتورهای جست و جو)، ادیت و ویرایش عکس، ساخت و تدوین کلیپ و تیزر، ساخت اینفوگرافیک، مدل‌سازی، ویرایش فایل‌های صوتی، آهنگسازی، پروژه‌های مهندسی، علوم تجربی، هنر، معماری، طراحی داخلی و... اشاره کرد.

۴.۱ کاربردها

فریلنسر یا آزادکار، شخصیست که با توجه به مهارت‌ها و تخصصی که دارد، به صورت اینترنتی و دورکاری، پروژه میگیرد و انجام می‌دهد و برای کارفرما می‌فرستد. فریلنسرها از طریق پروژه های دورکاری کسب درآمد می‌کنند. به این شیوه کاری فریلنسینگ گفته می شود. فریلنسرها، میتوانند از هر جایی که مایلند، حتی در خانه و هر ساعتی که دوست دارند، کار کنند و پروژه انجام دهند. فریلنسر ها ساعت کاری و محل کار ثابتی ندارند و خودشان محل و ساعات کاریشون رو انتخاب میکنند. فریلنسینگ نوعی سبک زندگی و کار است که امکان دورکاری و کسب درآمد از منزل رو برای فریلنسرها فراهم میکنه. بنابر بررسی‌ها تا سال ۲۰۲۵، نصف جمعیت شاغلان دنیا رو فریلنسرها تشکیل میدن. همچنین برونسپاری پروژه های طراحی وب و برنامه نویسی، تولید محتوا، طراحی گرافیک، تایپ، ترجمه و بسیاری مهارت های دیگه هم از این طریق امکان پذیر است.

۵.۱ مراحل کار

۶.۱ روش کار

۷.۱ ساختار گزارش

در این گزارش به پنج فصل زیر پرداخته شده است:

- در فصل اول به کلیات و اهداف پرداخته شده است.
- در فصل دوم به مفاهیم و دانش فنی پرداخته شده است.
- در فصل سوم به تحلیل نرم‌افزار پرداخته شده است.

- در فصل چهارم به طراحی نرم‌افزار پرداخته شده است.
- در فصل پنجم به جمع‌بندی و پیشنهادات پرداخته شده است.

فصل۲

مفاهیم و دانش فنی

مقدمه

در این فصل به مفاهیم و دانش فنی لازم به جهت درک نرم‌افزار پرداخته شده است.

BackEnd ۱.۲

Backend یا بک اند، به بخشی از یک وب سایت یا نرم افزار می‌گویند که برای کاربران قابل مشاهده نیست. به عبارتی دیگر هسته و مغز یک سایت است که وظیفه کنترل منطق آن را بر عهده دارد. سایت‌های دینامیک به برنامه نویس بک اند نیاز دارند تا منطق سایت را به وسیله زبان‌های برنامه نویسی پیاده‌سازی کنند. کاربران به کدهای نوشته‌شده در بک اند دسترسی ندارند و نمی توانند آن‌ها را مشاهده کنند. این بخش از سایت مانند قسمتی از کوه یخ است که در زیر سطح آب قرار گرفته است. سمت سرور با بخش سمت کاربر ارتباط مستقیم دارد و به اجزایی که در رابط کاربری طراحی شده‌اند جان می‌بخشد. برنامه نویس بک اند باید اطلاعات را متناسب با اهداف مختلف از پایگاه‌داده دریافت کند و در صورت نیاز پس از پردازش به کاربر نمایش دهد. بنابراین Backend از دو بخش منطق سایت و پایگاه داده تشکیل شده است.

JavaScript ۱.۱.۲

جاوا اسکریپت چیست؟

JavaScript که به اختصار JS نیز نامیده می‌شود، یکی از محبوبترین زبان‌های برنامه نویسی است. جاوا اسکریپت زبانی سطح بالا، داینامیک، شی‌گرا و تفسیری است که از شیوه‌های مختلف برنامه نویسی پشتیبانی می‌کند. از این زبان می‌توان برای برنامه نویسی سمت سرور ، (Server Side) اپلیکیشن‌های موبایل، بازی و اپلیکیشن‌های دسکتاپ استفاده کرد. بنابراین می‌توان اینگونه برداشت کرد که زبان برنامه نویسی جاوا اسکریپت ، یک زبان همه فن حریف است. اگر با هر یک از این اصطلاحات آشنایی ندارید نگران نباشید، زیرا در ادامه به توضیح هر یک از آن‌ها خواهیم پرداخت. برای اینکه بهتر متوجه چیستی زبان جاوا اسکریپت شوید، در ابتدا باید جواب سوالاتی مانند زبان کامپایلری چیست و چه تفاوتی با زبان مفسری دارد؟، زبان برنامه نویسی سمت سرور و سمت کاربر به چه نوع زبان‌هایی گفته می‌شود؟ را بدانید. پس از درک این مفاهیم می‌توانید آموزش جاوا اسکریپت را شروع کنید. همانطور که می‌دانید کامپیوترها تنها به زبان صفر و یک (Binary) صحبت می‌کنند و زبان دیگری را متوجه نمی‌شوند. ما در ابتدا برای برقراری ارتباط با ماشین‌ها سعی کردیم به زبان خود آنها، یعنی زبانی که به زبان صفر و یک نزدیک‌تر است، صحبت کنیم. به این نوع زبان‌ها که به صورت مستقیم با پردازنده در ارتباط‌اند، در اصطلاح، زبان‌های سطح پایین (Low Level) گفته می‌شود. از جمله این زبان‌ها می‌توان به اسمبلی اشاره کرد. اما یادگیری و تسلط به این زبان‌ها برای برنامه نویسان فوق العاده سخت بود. بنابراین متخصصین تصمیم به ساخت زبان‌هایی گرفتند که به زبان انسان‌ها نزدیک‌تر باشد. در اصطلاح به این زبان‌ها، زبان‌های سطح بالا (High Level) می‌گویند. زبانهای سطح بالایی مانند JavaScript کار را برای برنامهنویسان ساده‌تر کردند، زیرا ساختار نوشتاری و منطق آن‌ها بسیار به زبان انسان‌ها نزدیک‌تر شده است. پس می‌توان اینگونه نتیجه گرفت که آموزش جاوا اسکریپت نسبت به سایر زبان‌های برنامه نویسی سطح پایین ساده‌تر است. همانطور که دیدید در تعریف زبان برنامه نویسی جاوا اسکریپت به این نکته اشاره شد که این زبان از نوع زبان‌های مفسری است. برای درک ماهیت زبان‌های برنامه نویسی مفسری ابتدا فکر کنید که شما یک مترجم هستید. برای ترجمه یک متن، دو راه بیشتر ندارید. یا باید آنچه را دریافت می‌کنید به صورت خط به خط و همزمان ترجمه کنید، یا کل مطلب را یک جا ترجمه کنید. این دقیقا همان تفاوت میان زبان‌های مفسری (Interpreter) و زبان‌های کامپایلری

(Compiled) است.

تاریخچه زبان جاوا اسکریپت

جاوا اسکریپت اولین بار در می۱۹۹۵ در ۱۰ روز توسط برندن ایچ، یکی از کارکنان شرکت Netscape متولد شد! در ابتدا این شرکت به این نتیجه رسیده بود که به صفحات وب پویا و جذاب‌تری احتیاج دارد. این اولین قدم به سوی ساخت زبانی ساده بود. آقای براندان ایچ از طرف این شرکت مامور شد که زبانی اسکریپتی برای صفحات وب و دست بردن در کدهای HTML بسازد. ماموریت آقای ایچ این بود زبانی را ارائه کند که نه تنها متخصصان برنامه نویسی از آن استقبال کنند، بلکه به راحتی مورد استفاده طراحان هم باشد. این شرکت در ابتدا به فکر ارتقا و ساده سازی زبان Schema افتاد اما در نهایت به این نتیجه رسید که به زبانی شبیه جاوا اما با سینتکس ساده‌تر احتیاج دارد. در ابتدای کار اسم این زبان برنامه نویسی Mocha بود که بعد به Mona تغییر پیدا کرد. در سپتامبر همان سال اسم این زبان به LiveScript تغییر کرد و در آخر سریال تغییر اسم با انتخاب اسم JavaScript به اتمام رسید. نهایی شدن این اسم تنها به این دلیل بود که در آن روزها زبان برنامه نویسی Java بسیار پرطرفدار شده بود. انتخاب این نام برای این زبان بسیار هوشمندانه بود. زیرا در آن زمان این زبان با انتخاب این نام، توانست سهم زیادی از بازار جاوا را به خود اختصاص دهد. به هر حال در سال ۱۹۹۶ جاوا اسکریپت برای استاندارد شدن به سازمان ECMA سپرده شد. در نهایت اولین استاندارد جاوا اسکریپت با نام ECMAScript در سال ۱۹۹۷ منتشر شد. اولین اکما اسکریپت ECMA-۲۶۲ و آخرین ورژن آن با اسم VECMAScript 201 در ژوئن ۲۰۱۷ منتشر شد.

نقاط قوت زبان جاوا اسکریپت

هر یک از زبان هایی که در دنیای برنامه نویسی مورد استفاده قرار می‌گیرند نقاط قوت و ضعف هایی دارند که زبان جاوا اسکریپت هم از این موضوع مستثنا نیست. این زبان به دلیل مزایای فراوانی که دارد در میان برنامه نویسان از محبوبیت زیادی برخوردار است که به طور خلاصه به برخی از آنها اشاره می‌کنیم :

- بر اساس بررسی سایت stackoverflow محبوب‌ترین زبان برنامه نویسی سال ۲۰۱۸ است
- برای پردازش و اجرا به کامپایلر احتیاجی ندارد.
- یادگیری جاوا اسکریپت نسبت به خیلی از زبان‌های برنامه نویسی راحت‌تر است.
- به صورت کراس پلتفورم روی مرورگرها یا پلتفرم‌های مختلف اجرا می‌شود.
- نسبت به زبان‌های برنامه نویسی دیگر سبک‌تر و سریع‌تر است. فریم ورک ها،کتابخانه‌ها و به صورت کلی ابزارهای بسیار زیادی را در اختیارتان قرار می‌دهد.
- زبان بومی مرورگر وب است و در مرورگر کاربران پردازش می‌شود. امکان ایجاد صفحات وب تعاملی و پویا را به برنامه نویسان می‌دهد.
- در جواب عمل کاربران، عکس العمل نشان می‌دهد.

نقاط ضعف زبان جاوا اسکریپت چیست؟

برخی از ضعف‌های این زبان برنامه نویسی عبارتند از :

- دشواری در تشخیص دلیل خطا دادن و مشکل در دیباگ کردن
- محدودیت در اجرای اسکریپت‌های جاوا اسکریپت با ایجاد محدودیت هایی جهت حفظ امنیت
- اجرا نشدن بر روی مرورگرهای قدیمی
- نفوذپذیری نسبت به اکسپلویت‌ها و عوامل مخرب
- می تواند برای اجرای کدهای مخرب در کامپیوتر کاربران استفاده شود.
- با رندر شدن متفاوت بر روی ابزارهای مختلف می‌تواند باعث ایجاد تناقض و نداشتن یکپارچگی شود.

کاربرد جاوا اسکریپت

پیش‌تر به محبوبیت زبان جاوا اسکریپت اشاره کردیم. این محبوبیت بی دلیل نیست چرا که با این زبانِ شی گرا شما قادر خواهید بود تا سایت‌های بی‌روح خود را جان بخشی کنید و با کاربران خود تعامل داشته باشید. یعنی می‌توانید فایل‌های انیمیشنی، صوتی و تصویری را روی سایت خود به نمایش بگذارید. همچنین می‌توانید روی سایت‌تان تایمر قرار دهید، رنگ‌ها را با حرکت موس تغییر دهید و بسیاری کارهای دیگر که باعث جذابیت بیشتر صفحات وب می‌شوند. اما این تمام چیزی نیست که جاوا اسکریپت در اختیار شما قرار می‌دهد. شما با استفاده از این زبان می‌توانید شروع به ساخت برنامه‌های وب و موبایل و دسکتاپ کنید. برای این منظور می‌توانید از فریم‌ورک‌های مختلف JavaScript که مجموعه‌ای از کتابخانه‌ها را در اختیار شما قرار می‌دهند استفاده کنید. یکی از کارهای سرگرم کننده دیگری که می‌توانید از طریق این زبان انجام دهید، توسعه بازی‌های رایانه ای تحت مرورگر است. پس به صورت کلی می‌توان کاربردهای زبان جاوا اسکریپت را به صورت زیر بیان کرد :

- برنامه نویسی فرانت اند
- برنامه نویسی بک اند با جاوا اسکریپت
- برنامه نویسی نرم افزارهای موبایل
- برنامه نویسی نرم افزارهای دسکتاپ

NodeJS ۲.۱.۲

Node.js چیست ؟

Node.js یک پلتفرم سمت سرور مبتنی بر موتور جاوا اسکریپت گوگل کروم (V8 engine) می‌باشد. Node.JS تمام چیزهایی که برای اجرای یک برنامه نوشته شده به زبان جاوا اسکریپت را نیاز دارید برایتان فراهم می‌کند. آقای Ryan Dahl در سال 2009 Node.JS را معرفی کرد تا نشان دهد جاوا اسکریپت قدرتمندتر از این حرف‌ها است که فقط برای پویاسازی صفحات وب در فرانت اند استفاده شود. در واقع به کمک Node.js زبان برنامه نویسی جاوا اسکریپ به جای اجرا درمرورگر در

محیط سرور اجرا می‌شود. Node.js به شما اجازه می‌دهد به آسانی و سادگی برنامه‌های تحت شبکه مقیاس پذیر و بزرگ بنویسید. جاوا اسکریپت از سال ۱۹۹۵ در حال پیشرفت بود. هر چند این زبان تا مدت‌ها قبل حضور موفقی در سمت سرور نداشت و و تلاش هایی که توسط برنامه نویسان انجام شده بود، به مرور زمان از ذهن توسعه دهندگان دیگر محو می‌شد. تا اینکه با معرفی نود جی اس در سال ۲۰۰۹ مهره برگشت و به مرور زمان جاوا اسکریپت بیشتر و بیشتر در سمت سرور مورد استفاده قرار گرفت.

چرا باید از Node.js استفاده کنیم؟

Node.js بازدهی و انعطاف بالایی دارد نود در کنار V8 engine از زبان برنامه نویسی C++ استفاده کرده و سرعت بسیار بالایی دارد. هم V۸ هم Node.js به صورت مرتب آپدیت شده و با قابلیت‌های جدید جاوا اسکریپت هماهنگ می‌شوند، همینطور بازدهی آنها بالاتر رفته و مشکلات امنیتی آنها نیز برطرف می‌شود. همینطور به دلیل استفاده از زبان جاوا اسکریپت انتقال فایل JSON (متداول‌ترین قالب انتقال داده در وب) به طور پیش فرض بسیار سریع خواهد بود.

Node.js کراس پلتفرم است پلتفرم هایی مثل Electron.js یا NW.js به شما اجازه می‌دهند با نود جی اس برنامه‌های دسکتاپ بسازید. به این ترتیب می‌توانید برخی از کدهای برنامه تحت وب خود را در محیط ویندوز، لینوکس و مک اواس استفاده کنید. در واقع به کمک نود جی اس، همان تیمی که روی نسخه وب محصول کار می‌کنند، بدون نیاز به دانش تخصصی در زبان‌های C# یا Objective C یا سایر زبان هایی که برای ساخت برنامه‌های Native به کار می‌روند، می‌توانند یک برنامه دسکتاپ بسازند.

Node.js می‌تواند با میکروسرویس‌ها ترکیب شود اکثر پروژه‌های بزرگ در اول کار ساده بودند و در یک نسخه MVP معرفی شده بودند. اما به مرور زمان این سرویس‌ها بزرگتر شده و نیاز به اضافه کردن قابلیت‌های جدید در آنها حس می‌شد. گاهی وقت‌ها بزرگ شدن سرویس و اضافه کردن امکانات جدید به محصول می‌تواند برای تیم توسعه دهندگان تبدیل به یک کابوس شود. اما یک راه حل مناسب برای حل این مشکل استفاده از میکروسرویس است. میکروسرویس کمک می‌کند برنامه خود را بخش‌های کوچک تقسیم کنید که هر بخش می‌تواند توسط تیم متفاوت و حتی زبانی متفاوت نوشته شود. نود جی اس در کار با میکروسرویس‌ها عملکرد بسیار خوبی دارد.

Node.js چه کاربردهایی دارد؟

ساخت برنامه‌های تک صفحه ای (SPA) SPA مخفف single-page app بوده و برنامه هایی گفته می‌شود که تمام بخش‌های آن در یک صفحه پیاده سازی می‌شود. از SPA بیشتر برای ساخت شبکه‌های اجتماعی، سرویس‌های ایمیل، سایت‌های اشتراک ویدئو و غیره استفاده می‌شود. یکی از معروف‌ترین سایت هایی که به این شکل ساخته شده است، سرویس اشتراک ویدئو یوتیوب است. از آنجایی که نود جی اس از برنامه نویسی نامتقارن یا asynchronous به خوبی پشتیبانی می‌کند، برای ساخت برنامه‌های SPA انتخاب خوبی به حساب می‌آید

ساخت برنامه‌های RTA RTA مخفف real-time app می‌باشد. یعنی برنامه هایی که به صورت لحظه ای دارای تغییرات مختلفی هستند. به احتمال زیاد قبلا با این نوع برنامه‌ها کار کرده اید. برای مثال Google Sheets، Spreadsheets یا Slack از این دست برنامه‌ها هستند. در کل برنامه‌های تعاملی، ابزارهای مدیریت پروژه، کنفرانس‌های ویدئویی و صوتی و سایر برنامه‌های RTA عملیات‌های سنگین ورودی/خروجی انجام می‌دهند.

ساخت بازی‌های آنلاین تحت مرورگر وب ایده ساخت چت روم جذاب است، اما جذابیت آن زمانی بیشتر می‌شود که یک بازی هم برای مرورگر وب بنویسید و کنار آن بازی یک چت روم هم ارائه کنید. به کمک نود جی اس می‌توان به توسعه بازی تحت وب پرداخت. در واقع با ترکیب تکنولوژی‌های HTML۵ و ابزارهای جاوا اسکریپت (مثل Express.js یا Socket.io یا غیره) می‌توانید بازی‌های دوبعدی جذابی مثل Ancient Beast یا PaintWar بسازید.

NoSQL ۳.۱.۲

NoSQL چیست؟

در برنامه نویسی سنتی، پایگاه‌های داده معمولا از نوع SQL هستند؛ که یک پایگاه داده رابطه ای یا Relational است. پایگاه‌های داده رابطه ای ساده هستند و کار کردن با آن‌ها معمولا بی دردسر و راحت است. اما این نوع از پایگاه‌های داده یک مشکل بزرگ دارند. این مشکل زمانی خود را نشان داد که غول‌های نرم افزاری دنیا مثل گوگل، آمازون و فیسبوک احتیاج به تحلیلِ داده‌های با حجم و تعداد بالا یا همان Big Data پیدا کردند. پایگاه‌های داده رابطه ای به دلیل نوع ساختار خود، برای تحلیل داده‌های بزرگ غیر بهینه، ناکارا و همینطور کند بودند. البته در بعضی موارد هم استفاده از ساختار جدولی که در پایگاه‌های داده رابطه ای استفاده می‌شود تقریبا ناممکن بود. به همین دلیل ذخیره سازی حجم زیادی از داده‌های بی ساختار (Non-structured Data) سرعت و کارایی این پایگاه‌های داده را به شدت کاهش می‌داد. تا اینکه پایگاه‌های داده NoSQL پا به عرصه گذاشتند. پس همانطور که حدس می‌زنید، هدف اصلی ایجاد پایگاه‌های داده NoSQL کار با داده‌های بی ساختار و حجیم است. گفتیم که مشکل پایگاه‌های داده مبتنی بر SQL از نوع ساختار آن‌ها ناشی می‌شود. اما این ساختار چگونه است و چرا چنین مشکلاتی را ایجاد می‌کند؟ برای فهمیدن پاسخ این سوال احتیاج داریم کمی با ساختار پایگاه‌های داده SQL آشنا شویم.

پایگاه‌های داده NoSQL

پایگاه‌های داده NoSQL (Not Only SQL) برعکس نوع SQL از ساختارهای Schema غیر ثابت یا Schema Dynamic استفاده می‌کنند. این باعث می‌شود که برنامه نویسان احتیاجی به تشکیل ساختارهای سخت گیرانه مشخص، پیش از ایجاد پایگاه‌های داده را نداشته باشند. این پایگاه‌های داده می‌توانند انواع مختلفی داشته باشند و برعکس SQL برای ذخیره سازی داده‌ها از XML یا JSON استفاده می‌کنند. در ادامه انواع مختلفی از پایگاه‌های داده NoSQL را به شما معرفی می‌کنیم:

- پایگاه‌های داده کلید-مقدار یا Key-Value Database : در این نوع از پایگاه داده اطلاعات در قالب جفت‌های کلید-مقدار یا Key-Value ذخیره می‌شود. کلیدها نقش شناسه هر داده را بازی می‌کند. یعنی می‌توانیم با استفاده از آن‌ها مقادیر مختلف داده را ذخیره یا پیدا کنیم. پایگاه‌های داده کلید-مقدار به دلیل ساده بودن در کارکرد، پرکاربردترین نوع پایگاه‌های داده NoSQL هستند.

- پایگاه‌های داده ستونی یا Wide-Column Database : شاید تصور کنید پایگاه‌های داده ستونی همان پایگاه‌های داده رابطه ای هستند. اما این فقط ظاهر این گونه پایگاه‌های داده است که شبیه به نوع رابطه ای است. گفتیم که در پایگاه‌های داده رابطه ای لازم است که تعداد و نوع ویژگی‌های هر موجودیت و مقادیر داخل آن مشخص و ثابت باشد. این در حالی است که در پایگاه‌های داده ستونی، هر ستون در رکوردهای مختلف می‌تواند شامل داده هایی با ساختار و نوع متفاوت باشد.

- پایگاه‌های داده سندی یا Document Database : در این گونه پایگاه‌های داده برای ذخیره سازی داده‌ها از اسناد JSON یا XML استفاده می‌کنیم. پایگاه‌های داده سندی معمولا برای ذخیره سازی و استفاده از داده‌های پراکنده و بی ساختار استفاده می‌شوند.

- پایگاه‌های داده گرافی یا Graph Database : در این نوع از پایگاه‌های داده برای ذخیره سازی موجودیت‌ها و روابط بین آن‌ها از گراف استفاده می‌کنیم. پایگاه‌های داده گرافی برای مواردی که در آن‌ها به ایجاد ارتباط‌های متعدد بین جداول احتیاج داریم بسیار مناسب هستند.

- پایگاه‌های داده چند مدله یا Multimodel Database : پایگاه‌های داده چند مدله ترکیبی از انواع دیگر پایگاه داده هستند. در این نوع پایگاه‌های داده می‌توانیم داده‌ها را به روش‌های مختلفی ذخیره، و از آن‌ها استفاده کنیم.

مزیت‌های استفاده از NoSQL

پایگاه‌های داده NoSQL مزیت‌های بسیار زیادی دارند که آن‌ها را برای سیستم‌های بزرگ و توزیع شده تبدیل به بهترین گزینه می‌کند. به طور کلی می‌توان این مزیت‌ها را به این شکل خلاصه کرد:

- مقیاس پذیری بالا :(Scalability) پایگاه‌های داده NoSQL می‌توانند به راحتی با روش مقیاس پذیری افقی یا Scaling Horizontal گسترش پیدا کنند. این ویژگی باعث کم شدن پیچیدگی و هزینه مقیاس دادن به نرم افزار یا Scale کردن آن می‌شود.

- کارایی بالا :(Performance) در سیستم‌های توزیع شده NoSQL با تکثیر خودکار داده‌های NoSQL در سرورهای متعدد در سراسر دنیا، تاخیر در ارسال پاسخ از طرف سرور به پایین‌ترین حد ممکن می‌رسد.

- دسترسی بالا :(Availability) در سیستم‌های توزیع شده NoSQL به دلیل کپی شدن خودکار داده‌ها در سرورهای مختلف، با از دسترس خارج شدن یک یا چند سرور، پایگاه داده همچنان قابل دسترس و پاسخگو است.

MongoDB ۴.۱.۲

Mongo DB چیست؟

مونگو دیبی (Mongo DB) یکی از معروف‌ترین پایگاه داده‌های SQL No است که ساختار منعطفی دارد و بیشتر در پروژه هایی با حجم بالای داده استفاده می‌شود. این پایگاه داده پلتفرمی متن باز و رایگان است و با مدل داده‌های مستند گرا (Document - Oriented) کار می‌کند و در ویندوز، مکینتاش و لینوکس قابل استفاده است. مقادیر داده ای ذخیره شده در مونگو دیبی، با دو کلید اولیه (Primary Key) و ثانویه (Secondary Key) مورد استفاده قرار می‌گیرند. مونگو دیبی شامل مجموعه ای از مقادیر است. این مقادیر به صورت سندهایی (Document) هستند که با اندازه‌های مختلف، انواع مختلفی از داده‌ها را در خود جای داده اند. این مسئله باعث شده که مونگو دیبی بتواند داده هایی با ساختار پیچیده مانند داده‌های سلسله مراتبی و یا آرایه ای را در خود ذخیره کند.

ویژگی‌های Mongo DB

- مونگو دیبی به علت مستند گرا بودن مدل ذخیره داده‌ها در مقایسه با دیتابیس‌های رابطه ای بسیار منعطف‌تر و مقیاس پذیرتر است و بسیاری از نیازمندی‌های کسب و کارها را برطرف می‌کند.

- این پایگاه داده برای تقسیم داده‌ها و مدیریت بهتر سیستم از شاردینگ (Sharding) استفاده می‌کند. شاردینگ به معنی تکه تکه کردن است و در لود بالای شبکه انجام می‌شود. به گونه ای که دیتابیس به چند زیربخش تقسیم می‌شود تا روند پاسخ دهی به درخواست هایی که از سمت سرور می‌آید، راحت‌تر شود.

- داده‌ها با دو کلید اولیه و ثانویه قابل دسترسی هستند و هر فیلدی قابلیت کلید شدن را دارد. این امر زمان دسترسی و پردازش داده را بسیار سریع می‌کند.

- همانند سازی (Replication) یکی دیگر از خصوصیات مهم مونگو دیبی است. در این تکنیک از یک داده به عنوان داده اصلی کپی هایی تهیه شده و بخش‌های دیگری از سیستم پایگاه داده ذخیره می‌شود. در صورت از بین رفتن و یا مخدوش شدن این داده، داده‌های کپی شده به عنوان داده اصلی و جایگزین مورد استفاده قرار می‌گیرند.

روش کار Mongo DB

در دیتابیس‌های رابطه ای داده‌ها به شکل رکورد (Record) نگهداری می‌شوند اما در مونگو دیبی، ساختار نگهداری داده‌ها به شکل سند است. هر سند از نوع Binary JSON یا BSON است و دارای فیلدهای کلید و مقدار می‌باشد. برای اجرا کردن کدهایی که در مونگو دیبی نوشته شده است باید از طریق Mongo Shell اقدام کرد. مونگو شل رابط تعاملی دیتابیس و برنامه نویس محسوب می‌شود و به آن‌ها اجازه ارسال کوئری (Query) و به روزرسانی داده‌ها را می‌دهد.

مزایا و معایب Mongo DB

دیتابیس‌های رابطه ای دارای اسکیما (Schema) هستند. یعنی ساختار خاصی برای داده هادر نظر گرفته و مدل‌های محدودی را ذخیره می‌کنند. اما مونگو دیبی و به طور کلی دیتابیس‌های NoSQL در برابر پذیرش داده هایی با توع مختلف بسیار منعطف هستند و این مزیت مهمی برای برنامه نویسان محسوب می‌شود. مقیاس پذیری این پایگاه داده باعث استفاده از آن در پروژه هایی می‌شود که با کلان داده‌ها (Big Data) سروکار دارند. علاوه بر مزایای گفته شده مشکلاتی نیز در مونگو دیبی وجود دارد که ممکن است دردسرساز شود. این دیتابیس در استفاده از کلید خارجی (Foreign Key) برای داده‌ها ضعف دارد و ممکن است پایداری داده‌ها و یکپارچگی سیستم را به هم بریزد. همچنین در خوشه بندی داده‌های موجود در این پایگاه داده، تنها می‌توان یک گره (Node) را به عنوان گره اصلی (Master) انتخاب کرد که اگر از بین برود، ممکن است مرتب سازی زیرگره‌های آن از بین برود. این مشکل در پایگاه داده کاساندرا (Cassandra) برطرف شده است.

FrontEnd ۲.۲

فرانت اند یا Front End ، به بخش قابل مشاهده‌ی یک وب سایت یا نرم افزار توسط کاربران می‌گویند. فرانت اند، کدهای غیر قابل فهم برای کاربران را در قالب ظاهری گرافیکی و بصری به آن‌ها نمایش می‌دهد تا بتوانند به راحتی از بخش‌های مختلف سایت استفاده کنند. در این بخش، فرم‌های ورودی اطلاعات، صداها، تصاویر، ویدئوها و به صورت کلی هر چیز دیگری که برای کاربر قابل درک باشد، قرار می‌گیرد. فرانت اند به دو بخش اصلی طراحی و توسعه رابط کاربری تقسیم می‌شود. در بخش طراحی، طراحان با نرم افزارهای گرافیکی مانند فتوشاپ، ادوبی ایکس دی، فیگما و... ظاهر سایت را طراحی می‌کنند. اما بخش توسعه‌ی رابط کاربری مربوط به پیاده سازی ظاهر سایت در قالب کدهای HTML ،CSS, JS است. بخش قابل مشاهده سایت برای کاربران در سمت فرانت را سمت کاربر یا Client Side می‌نامند. بنابراین کدهای نوشته شده در سمت فرانت اند، در مرورگر کاربر پردازش و اجرا می‌شوند. یعنی کاربر به راحتی به این کدها دسترسی مستقیم دارد و می‌تواند

آن‌ها را مشاهده کند. فرانت اند (Front-end) با بخش بک اند (Back-end) در ارتباط مستقیم است و بر روی تجربه کاربران هنگام استفاده از محصول تاثیر بسیاری می‌گذارد.

Html ۱.۲.۲

HTML چیست؟

HTML مخفف Hyper Text Markup Language بوده و در فارسی به آن زبان نشانه گذاری ابرمتن می‌گویند. دقت کنید که HTML یک زبان برنامه نویسی نیست، بلکه یک زبان نشان‌گذاری یا Markup language به حساب می‌آید.

تاریخچه زبان HTML

برای اینکه بدانیم HTML از کجا آمده باید سفر کوتاهی به سال ۱۹۹۱ داشته باشیم. زمانی که آقای Tim Berners-Lee کار خود را روی 18 Tag یا همان برچسب ساده شروع کرد و اولین نسخه HTML را طراحی کرد. HTML روز به روز پیشرفت کرد و در هر نسخه امکانات بیشتری را در قالب تگ‌های کاربردی‌تر در اختیار طراحان قرار داد. به این ترتیب این زبان مشکلات قبلی خود را به مرور رفع کرد. HTML۴ در سال ۱۹۹۹ معرفی شد و توانست تا مدت‌ها توسط طراحان وب مورد استفاده قرار گیرد، تا این که بزرگترین تحول تاریخ HTML با معرفی HTML۵ اتفاق افتاد. این نسخه از زبان HTML توانست بیش از پیش به توسعه دهندگان در طراحی سایت‌ها کمک کند که در ادامه می‌خواهیم با آن بیشتر آشنا شویم.

HTML چطور کار می‌کند؟

HTML عناصر مختلفی را از جمله پاراگراف، لیست، عکس، صوت و غیره کنار هم قرار می‌دهد تا چهارچوب اصلی صفحه وب را ایجاد کند. به زبان ساده‌تر ما با HTML بدنه اصلی صفحه وب را می‌سازیم. اگر HTML را شبیه به یک ساختمان در حال ساخت در نظر بگیریم، مهندس عمران که پی ساختمان را ریخته و اسکلت آن را می‌سازد حکم کسی را دارد که ساختار اصلی صفحات وب را با HTML می‌سازد. همچنین مهندس معماری که وظیفه دارد ظاهر ساختمان را زیباتر کند مانند کسی است که به کدنویسی با CSS می‌پردازد. البته در دنیای وب معمولا وظیفه کدنویسی HTML و CSS به عهده یک نفر خواهد بود. فایل‌های HTML با پسوند htm. یا html. در سیستم ذخیره می‌شوند. این فایل‌ها تقریبا توسط همه مرورگرهای وب پشتیبانی می‌شوند و به راحتی می‌توانند محتویات آن را رندر کنند. منظور از رندر کردن این است که عناصر داخل سایت که ترکیبی از کد، تصویر، انیمیشن، ویدئو یا غیره هستند، تبدیل به اطلاعات قابل نمایش برای کاربران می‌شوند.

تگ چیست؟

HTML به کمک برچسب‌ها (Tags) عناصر مختلف را کنار هم می‌چیند و هر کاربر با توجه به نیاز خود از آن‌ها استفاده می‌کند. شاید بپرسید تگ چیست؟ تگ‌ها عناصری هستند که وظایف گوناگونی دارند و با فراخوانی هر کدام کارشان شروع شده و با بستن تگ کارشان تمام می‌شود. مثلا برای نوشتن پاراگراف‌ها در زبان HTML از تگ p استفاده می‌شود و زمانی که پاراگراف تمام شده، تگ هم بسته می‌شود. همچنین برای نشان دادن لینک‌ها از تگ a استفاده در صفحات وب استفاده می‌شود. تگ‌های HTML در حقیقت همان دستورالعمل‌های این زبان هستند که به مرورگر می‌گویند صفحه مورد نظر از چه عناصری تشکیل شده است. هر کدام از این Tag معنا و مفهوم خاصی دارند و به شما امکاناتی مانند تغییر شکل ظاهری متن‌ها، ساخت لیست‌های مختلف و به هم متصل کردن صفحات را می‌دهند. همچنین از آن‌ها برای کار با صدا، تصویر و غیره استفاده می‌شود.

چرا HTML یک زبان برنامه نویسی نیست؟

HTML هرگز نمی‌تواند یک زبان برنامه‌نویسی باشد. زیرا اصلا ویژگی‌های یک زبان برنامه‌نویسی، مثل متغیرها، توابع، شرط‌ها، حلقه‌ها و... را ندارد. پس کاملا اشتباه است اگر HTML را یک زبان برنامه‌نویسی بدانیم. می‌توانیم درباره‌ی HTML بگوییم که ابزاری است که با استفاده از تگ‌ها، می‌تواند صفحات وب را برای ما ساختاردهی کند.

مزایا و معایب زبان HTML چیست؟

HTML در کنار css و js هسته اصلی وب را تشکیل می‌دهد و یک زبان بسیار مهم در دنیای وب حساب می‌شود. این زبان مزیت‌ها و محدودیت‌هایی هم دارد که در ادامه به آن‌ها اشاره می‌کنیم و می‌بینیم دلیل اصلی ماندگاری HTML چیست و چرا این زبان با تمام مشکلاتش هنوز زبان شماره یک وب به حساب می‌آید. برخی از مهمترین مزایا و معایب این زبان عبارتند از:

مزایای HTML :

- یادگیری آسان و لذت‌بخش
- قابلیت اجرا در تمام مرورگرها
- متن باز و رایگان بودن
- ادغام آسان با زبان‌های سمت سرور مثل php

معایب HTML :

- استاتیک بودن و وابستگی به زبان‌های سمت سرور برای تعامل با کاربر
- ضعف در پشتیبانی از مرورگرهای قدیمی
- نیاز به طراحی جداگانه هر صفحه به دلیل نبود قواعد منطقی برنامه نویسی

Css ۲.۲.۲

css چیست؟

سی اس اس مخفف Cascading Style Sheet (CSS) است. زبان css یک زبان طراحی صفحات وب برای ایجاد و ساخت مشخصات ظاهری اسناد و اطلاعات وب سایت می باشد. css یکی از رایج ترین و محبوب ترین ابزارهای طراحی صفحات وب سایت نوشته شده توسط زبان HTML و یا XHTML می باشد و همچنین از زبان های اسکریپت دیگری مانند ، SVG plain XML و XUL نیز به خوبی پشتیبانی می نماید.

در کدنویسی با استفاده از CSS می‌توانید استایل سایت مثل رنگ، فونت، تصاویر پس زمینه و ... را بصورت دلخواه تغییر دهید.

هدف و کاربرد css چیست ؟

هدف از تولید css در واقع جداسازی اطلاعات محتوا (که توسط زبانی مانند HTML نوشته شده اند) از اطلاعات ظاهری مانند صفحه بندی، رنگ و سایز و نوع فونت می باشد. این جداسازی موجب افزایش سرعت در دسترسی به سایت، انعطاف پذیری بیشتر برای کنترل ویژگی های ظاهری، قابلیت طراحی چندین صفحه با یک فرمت یکسان و جلوگیری از پیچیدگی و انجام کارهای تکراری در طراحی وب سایت می گردد.

از دیگر کاربرد css این است که می توان تنظیماتی را اعمال نمود که نمایش صفحه وب سایت مورد نظر بسته به اندازه صفحه نمایش کاربر متغیر باشد که به آن اصطلاحا طراحی ریسپانسیو می گویند. در صورتی که مدیر وب سایت چندین نوع نمایش را برای یک صفحه وب سایت خود تنظیم نموده باشد، css برای تصمیم گیری اینکه کدام حالت را به نمایش بگذارد، از ابزارهای تعیین اولویت استفاده می نماید.

مزایای سی اس اس چیست ؟

- سازگاری بیشتر در طراحی
- گزینه های قالب بندی بیشتر
- کد سبک
- بارگیری سریعتر
- بهینه سازی موتور جستجو
- دسترسی بهتر به کد

معایب سی اس اس چیست ؟

- کنترل ضعیف صفحه بندی های قابل انعطاف
- عدم امکان انتخاب گزینه های والد
- محدودیت در کنترل فرم های عمودی
- عدم وجود توضیحات لازم در زبان css
- بروز مشکلاتی در ساخت ستون ها

Scss ۳.۲.۲

Scss چیست ؟

Scss پسوند نحوی CSS است. این بدان معنی است که هر شیوه نامه معتبر CSS یک پرونده SCSS معتبر با همان معنی است. علاوه بر این ، SCSS بیشتر هک های CSS و نحو اختصاصی فروشنده ، مانند فیلتر قدیمی IE را می فهمد.

Figma ۴.۲.۲

فیگما چیست؟

فیگما یک برنامه‌ی طراحی رابط کاربری است که روند ساخت آن در سال ۲۰۱۲ شروع شد و نهایتا در سال ۲۰۱۶ اولین نسخه از آن در اختیار عموم قرار گرفت. فیگما در حالت کلی با دیگر ابزارهای ویرایشی متفاوت است؛ زیرا یک ابزار تحت وب (web based) است: یعنی به صورت مستقیم در مرورگر اجرا می‌شود. این به این معنی است که شما می‌توانید در هر زمانی به پروژه‌های خود دسترسی پیدا کنید و بدون نیاز به خرید مجوز و یا نصب نرم‌افزار از هر کامپیوتری و یا پلتفرمی کار با آن را شروع کنید. نیازی نیست نگران حجم مصرفی اینترنت خود باشید، چراکه فیگما مصرف پهنای باند زیادی ندارد. با این حال، کاربران ویندوز و مک می‌توانند به راحتی فیگما را روی سیستم عامل خود نصب کرده و از برخی از امکانات آن به صورت آفلاین استفاده کنند.

فیگما یک برنامه‌ی رایگان است و در آن شما می‌توانید تا حتی سه پروژه‌ی فعال را به صورت همزمان ایجاد و ذخیره کنید. این سیاست به شما فرصت می‌دهد تا کار با نرم افزار را یاد بگیرید، آزمایش کنید و روی پروژه‌های کوچک مشغول به طراحی شوید. همچنین فارسی نوشتن در فیگما (در محیط ویرایشگر) بدون هیچ مشکلی امکان پذیر است که خبر خوشی برای طراحانی است که نیازمند استفاده از زبان فارسی هستند.

رابط کاربری (User Interface) ، یک عضو جداناپذیر از عوامل موثر بر بازاریابی و ارتباط کاربران با کسب و کارهای امروزی به شمار می‌رود. یک رابط کاربری خوب در اغلب موارد با نیازهای کاربران سنجیده می‌شود و اگر به درستی طراحی شود، کاربر برای کار با آن نیازی به آموزش ندارد و از کار کردن با آن لذت می‌برد. به همین دلیل نرم افزارهای متعددی برای ساخت یک رابط کاربری مناسب به وجود آمده‌اند.

یکی از برنامه‌های طراحی رابط کاربری، فیگما (Figma) است. با استفاده از فیگما، می‌توان یک رابط کاربری را به راحتی و در زمانی بسیار کم، به گونه‌ای طراحی کرد که به اصطلاح User Friendly باشد.

ویژگی‌های فیگما

- **استفاده از کامپوننت (Components)** : ساخت و ترکیب چند شی با یکدیگر (مشابه با نرم افزار Sketch)
- **ساخت نمونه‌ی اولیه یا پروتوتایپ** : قابلیت کلیک کردن روی کامپونت‌های مختلف در آن (مشابه با نرم افزار InVision)
- **کامنت گذاری هنگام طراحی** : اضافه کردن، نشانه‌گذاری و پاسخ دادن نظرات توسط اعضای تیم در حال کار روی پروژه
- **برنامه نویسی بدون کد** : قابلیت دریافت ابعاد و ویژگی‌ها در قالب فایل استایل، دانلود آیکون‌ها و عکس‌ها را از URL پروژه (مشابه با نرم افزار Zeplin)
- **تنظیم فضاها** : ساخت کامپونت با قابلیت تغییر اندازه با استفاده از تنظیمات فضاها (مشابه با نرم افزار Sketch)
- **مدیریت پروژه با Version Control** : مشاهده‌ی تاریخچه و بازگشت به یک نسخه‌ی خاص توسط اعضای تیم
- **کار همزمان توسط چند نفر** : مشاهده‌ی نشانگر موس سایر اعضای تیم هنگام کار بر صفحه (هرچند به طور کلی این کار توصیه نمی‌شود.)

- **پلاگین‌ها** : استفاده از فونت‌ها، استایل‌های سایر کاربران در قالب پلاگین‌های کاربردی

۳.۲ الگوی معماری MVC

۱.۳.۲ MVC چیست ؟

معماری mvc یک نوع استاندارد کد نویسی برای طراحی سایت حرفه ای با زبان های مختلف مثل php، asp.net و ... می باشد ، mvc در واقع مخفف کلمه‌های model، view، controller می‌باشد. کلمه mvc مخفف model، view، controller می باشد و وظیفه اصلی آن به زبان ساده و مختصر جدا کردن بخش های منطقی برنامه از بخش های سمت کاربر می باشد ، بدین ترتیب تمامی بخش های منطقی از سمت کاربر جدا شده و در نتیجه انجام تغییرات و توسعه دادن یک سایت یا یک سیستم تحت وب برای تیم برنامه نویسی بسیار راحت تر قابل انجام خواهد بود.

۲.۳.۲ تاریخچه MVC

الگوی معماری MVC ابتدا در اواخر دهه ۱۳۵۰ شمسی توسط تریگو رینسکویگ (Trygve Reenskaug) مطرح شد. اولین گزارش راجع‌به MVC زمانی نوشته شد که رینسکویگ با دانشمندی در آزمایشگاه علوم و تحقیقات زیراکس (Zerox PARC) در سال ۱۳۵۷ دیدار می‌کرد. در ابتدا، MVC با نام «Thing Model View Editor» شناخته می‌شد، اما خیلی زود نام آن به «Model View Controller» تغییر داده شد. هدف تریگو این بود که مشکل کاربران در مدیریت یک پایگاه‌داده بزرگ و پیچیده را برطرف کند. کاربرد MVC در طول سال‌ها تغییر کرده است. به دلیل اینکه MVC قبل از مرورگرهای وب اختراع شده است، در ابتدا به عنوان یک الگوی معماری برای رابط کاربری گرافیکی (GUI) مورد استفاده قرار می‌گرفت. مدل MVC اولین بار در اواسط دهه شصت شمسی در زبان برنامه‌نویسی Smalltalk ارائه شد. سپس، الگوی معماری MVC برای اولین بار به عنوان یک مفهوم عمومی به صورت یک مقاله در سال ۱۳۶۷ پذیرفته شد. پس از مدتی، الگوی معماری MVC به میزان زیادی در اپلیکیشن‌های تحت وب مورد استفاده قرار گرفت.

۳.۳.۲ ویژگی های MVC

- آزمایش‌پذیری ساده و بدون دردسر؛ یک فریم‌ورک قابل آزمایش، گسترش‌پذیر و قابل شخصی‌سازی به میزان زیاد.
- امکان مدیریت و کنترل کامل روی HTML و همچنین هاURL در الگوی معماری MVC
- بهره‌مندی از ویژگی‌های موجود در JSP Django، ASP.NET، و سایر موارد در MVC
- تفکیک بسیار واضح منطق وظایف اپلیکیشن به صورت مدل، نما و Controller در الگوی معماری MVC
- مسیریابی URL برای URLهای سازگار با سئو (SEO) و نگاشت URL قدرتمند برای URLهای قابل درک و قابل جستجو
- پشتیبانی از توسعه آزمون‌محور (Test Driven Development | TDD)

۴.۳.۲ اجزای MVC

الگوی معماری MVC در بخش‌های قبلی مطلب «MVC چیست» به طور ساده بیان شد، اکنون در این بخش، سه لایه مهم الگوی معماری MVC با جزئیات بیش‌تری مورد بررسی قرار خواهند گرفت. همان‌طور که بیان شد، سه جزء الگوی معماری MVC به شرح زیر است.

- مدل :(Model) این لایه شامل تمام داده‌ها و منطق مرتبط با آن داده‌ها است.
- کنترل‌گر :(Controller) یک رابط میان قطعات مدل و نما به حساب می‌آید.
- نما :(View) داده‌ها را به کاربر نمایش و ارائه می‌دهد یا در واقع تعاملات کاربر را مدیریت می‌کند.

در ادامه، هر یک از این قطعات و اجزاء الگوی معماری MVC به طور جداگانه شرح داده شده است.

مدل (Model)

مدل به عنوان پایین‌ترین سطح در ساختار معماری برنامه کاربردی شناخته می‌شود. این مسئله، به این معنا است که لایه مدل مسئولیت نگهداری و مدیریت منطقی داده‌ها را بر عهده دارد. مدل در واقع به پایگاه‌داده متصل است. بنابراین، هر کاری که با داده‌ها انجام می‌شود، از قبیل افزودن یا دریافت داده‌ها در بخش مدل انجام می‌شود. قطعه یا لایه مدل، داده‌ها و منطق مربوط به آن‌ها را ذخیره‌سازی می‌کند. مدل به درخواست‌های Controller پاسخ می‌دهد. زیرا Controller هیچ‌گاه به طور مستقیم و به خودی خود با پایگاه‌داده در ارتباط نیست. مدل با پایگاه‌داده مکاتبه می‌کند و سپس داده‌های مورد نیاز را به Controller انتقال می‌دهد. همچنین، مدل هیچ وقت به صورت مستقیم با لایه نما ارتباط ندارد.

کنترل‌گر (Controller)

Controller نقش اصلی را ایفا می‌کند. زیرا Controller بخشی است که امکان ارتباط دو طرفه میان مدل و نما را به وجود می‌آورد. بنابراین، Controller نقش یک میانجی را دارد. Controller نیازی به دخالت در مدیریت منطق داده‌ها ندارد و فقط آنچه باید انجام شود را به مدل انتقال می‌دهد. پس از دریافت داده‌ها از مدل، Controller آن‌ها را پردازش می‌کند و سپس تمام آن اطلاعات را برداشته و به نما ارسال می‌کند و نحوه نمایش آن‌ها را به نما توضیح می‌دهد. باید توجه داشت که نما و مدل مستقیماً امکان مکاتبه و ارتباط ندارند.

نما (View)

بازنمایی داده به وسیله قطعه یا لایه نما (View) انجام می‌شود. نما در واقع برای کاربر یک رابط کاربری یا UI تولید می‌کند. بنابراین، در خصوص کاربردهای وب، می‌توان قطعه نما را همان بخش HTML و CSS در نظر گرفت. نماها به وسیله داده‌ها ساخته می‌شوند و داده‌ها توسط قطعه مدل گردآوری می‌شوند اما، این داده‌ها به صورت مستقیم از مدل به نما انتقال داده نمی‌شوند، بلکه این کار از طریق Controller صورت می‌گیرد

۵.۳.۲ مزایای MVC

در این بخش از مطلب فواید اساسی استفاده از الگوی معماری MVC فهرست شده است.

- نگهداری کد (Code maintenance) ، تعمیم و رشد آن در یک چارچوب مبتنی بر الگوی معماری MVC بسیار ساده‌تر است.
- قطعات در الگوی MVC قابل استفاده مجدد هستند.
- یک قطعه مبتنی بر الگوی معماری MVC را می‌توان به طور مستقل از کاربر آزمایش کرد.
- پشتیبانی آسان برای مشتریان جدید، به واسطه الگوی معماری MVC امکان‌پذیر می‌شود.
- در الگوی معماری MVC توسعه قطعات مختلف را می‌توان به صورت موازی انجام داد.
- الگوی معماری MVC با تقسیم‌بندی یک اپلیکیشن به سه واحد مستقل نما، مدل و Controller از بروز پیچیدگی جلوگیری می‌کند.
- الگوی معماری MVC تنها از یک الگوی Front Controller استفاده می‌کند. این الگو، درخواست‌های ایجاد شده در وب‌اپلیکیشن را تنها از طریق یک Controller پردازش می‌کند.
- الگوی معماری MVC بهترین پشتیبانی را برای توسعه آزمون‌محور (Test-Driven Development) ارائه می‌دهد.
- الگوی MVC با وب‌اپلیکیشن‌های پشتیبانی شده توسط گروه‌های بزرگ طراحان و توسعه‌دهندگان، به خوبی کار می‌کند.
- تفکیک دغدغه‌ها (Clean Separation of Concerns | SoC) در MVC واضح و شفاف است.
- الگوی معماری MVC با بهینه‌سازی موتور جستجو (سئو | (SEO سازگاری دارد.
- تمام کلاس‌ها و اشیاء مستقل از یکدیگر هستند تا بتوان هر یک را به طور مجزا آزمایش کرد.
- الگوی معماری MVC امکان گروه‌بندی منطقی اعمال مرتبط با یکدیگر در یک Controller را فراهم می‌کند.

۶.۳.۲ معایب MVC

در ادامه این بخش از مطلب، معایب MVC و نقاط ضعف آن فهرست شده است.

- پیچیدگی MVC بالاست.
- برای اپلیکیشن‌های کوچک کارایی ندارد و مناسب نیست. دسترسی به داده‌ها در لایه نما کارایی لازم را ندارد.
- خواندن، تغییر دادن، آزمایش واحدها و استفاده مجدد از یک برنامه توسعه داده شده مبتنی بر الگوی معماری MVC دشوار است.
- ناوبری فریم‌ورک MVC ممکن است گاهی با اضافه شدن لایه‌های انتزاعی جدید پیچیده شود. این مسئله نیازمند این است که کاربران با معیارهای تجزیه MVC تطبیق دهند.
- پشتیبانی رسمی از تایید اعتبار در MVC وجود ندارد.

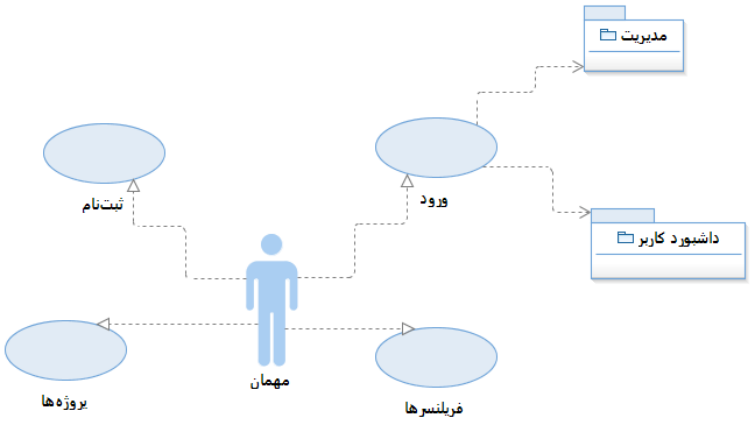
- پیچیدگی و عدم کارایی داده‌ها در MVC
- استفاده از الگوی معماری MVC با رابط کاربری امروزی دشوار است.
- برای اجرای برنامه‌نویسی موازی به چندین برنامه‌نویس نیاز است.
- در الگوی معماری MVC نیاز به دانش در چندین حوزه فناوری وجود دارد.
- نگهداری از حجم بالای کدها در لایه Controller چالش ایجاد می‌کند.

فصل۳

تجزیه و تحلیل نرم‌افزار

مقدمه

در این فصل به دیاگرام‌ها و سناریوهای نرم‌افزار پرداخته شده است.



شکل ۱.۳: دیاگرام UC ساختار کلی

۱.۳ ثبت نام

مورد استفاده: ثبت‌نام

شرح مختصر UC: این قسمت مهمان در سایت ثبت‌نام می‌کند.

پیش شرط: دانشجو باشد.

سناریو اصلی:

۱. شروع

۲. مهمان دکمه ثبت‌نام را انتخاب می‌کند و سیستم فرم ثبت‌نام را به مهمان نمایش می‌دهد.

۳. مهمان فرم را تکمیل می‌کند و با دکمه ارسال، فرم تکمیل شده را به سیستم ارسال می‌کند.

۴. سیستم فرم ثبت‌نام را بررسی می‌کند و اطلاعات فرم را در بانک اطلاعات ثبت می‌کند.

۵. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در اطلاعات وارد شده

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در اطلاعات وارد شده فرم ثبت‌نام اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به مهمان نمایش داده می‌شود و درخواست اصلاح اطلاعات وارد شده را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: کاربر با موفقیت ایجاد شود.
شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت آمیز بودن ثبت‌نام اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و یک پیغام به مهمان نمایش داده می‌شود که اطلاعات با موفقیت ثبت و کاربر ایجاد شده است.

۳. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

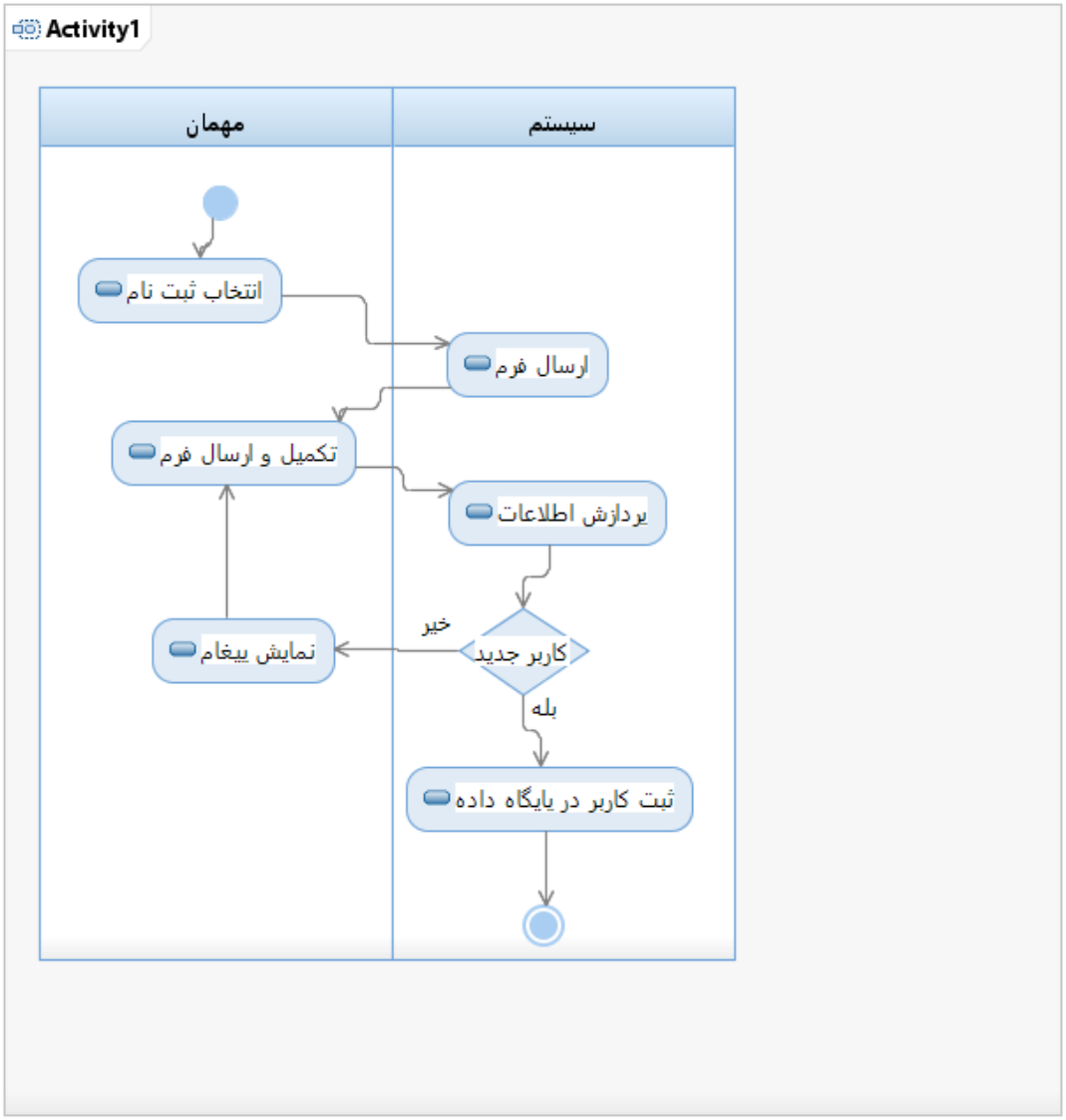
سناریو فرعی ۳: کاربر موجود باشد.
شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت وجود کاربر در بانک اطلاعات اجرا می‌شود.

۱. شروع

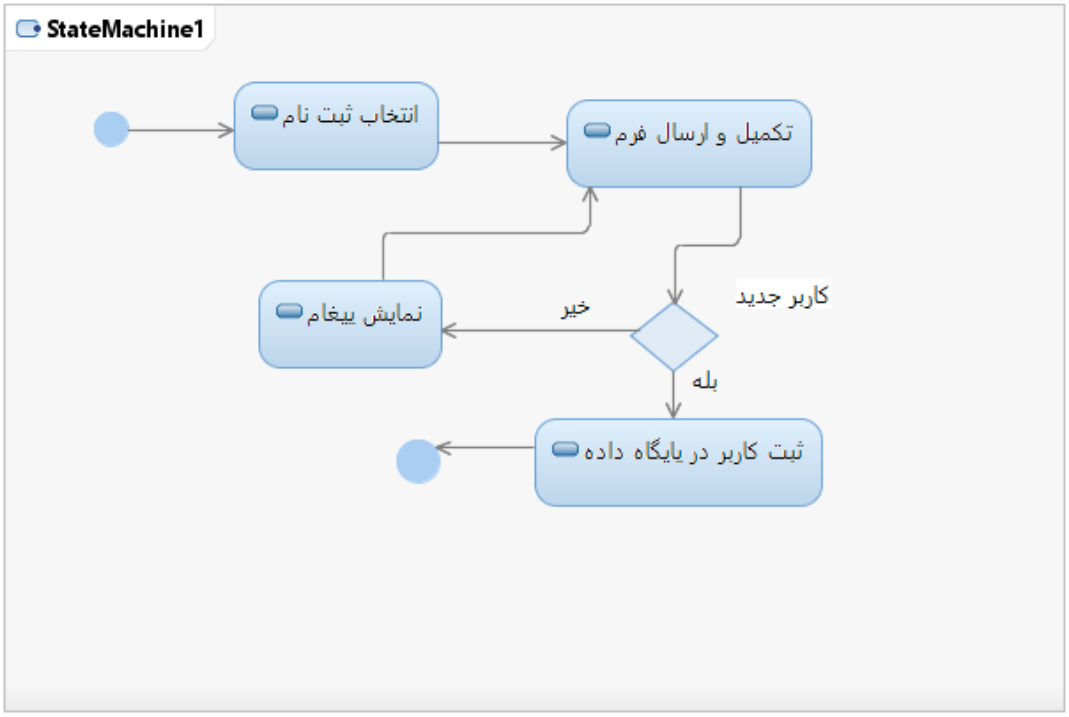
۲. یک پیغام به مهمان نمایش داده می‌شود که اطلاعات کاربری از قبل وجود دارد و دکمه ورود به سایت نمایش داده می‌شود.

۳. پایان

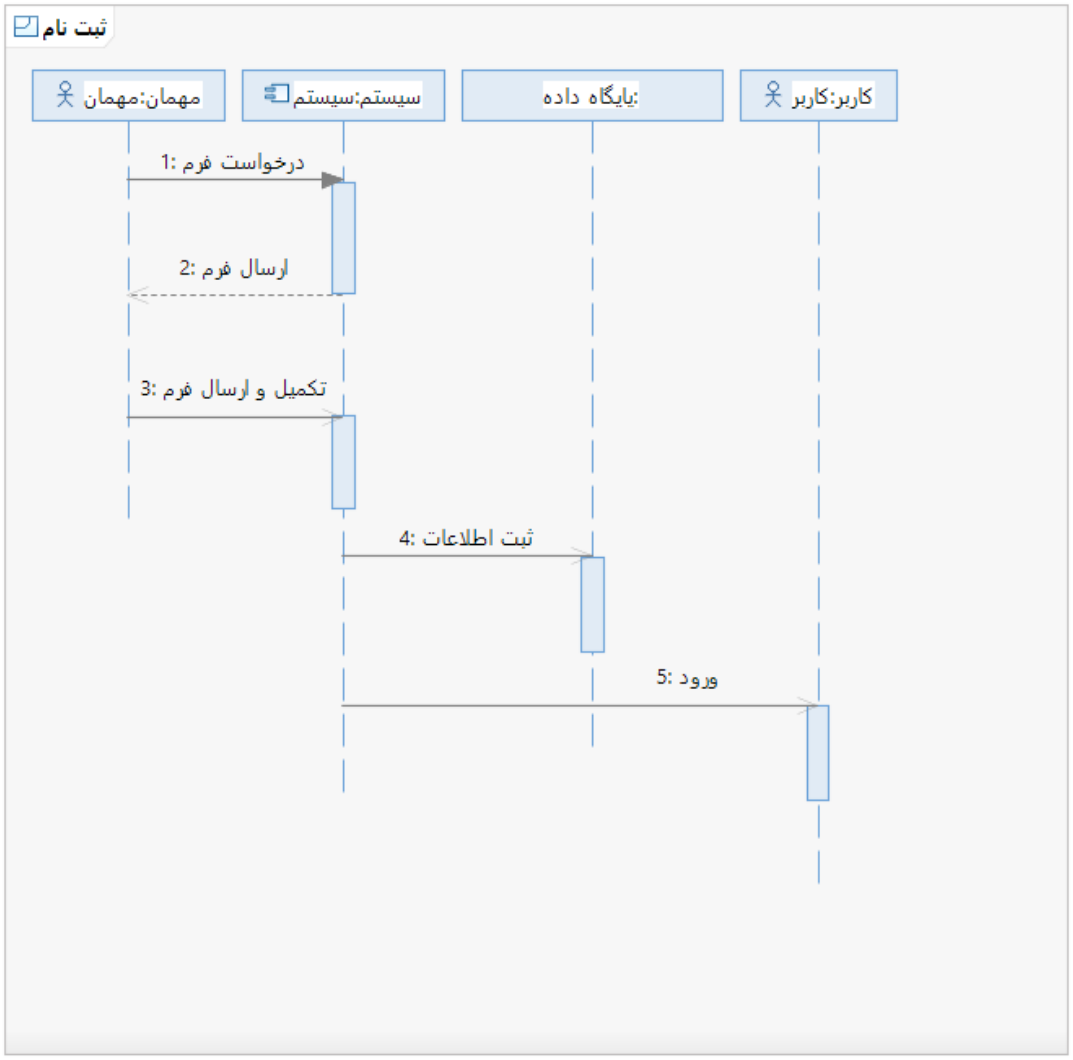
پس شرط: ندارد .



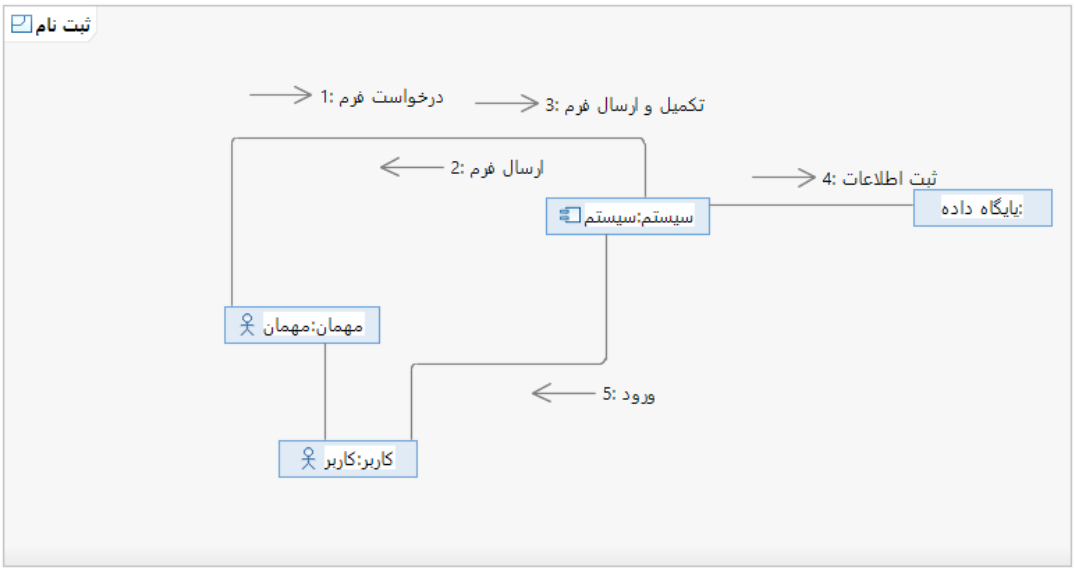
شکل ۲.۳: دیاگرام فعالیت ثبت‌نام



شکل ۳.۳: دیاگرام حالت ماشین ثبت‌نام



شکل ۴.۳: دیاگرام توالی ثبت‌نام



شکل ۵.۳: دیاگرام همکار ثبت‌نام

۲.۳ ورود

مورد استفاده: ورود

شرح مختصر UC: در این قسمت مهمان با عنوان کاربر وارد می‌شود.

پیش شرط: ثبت‌نام انجام شده باشد.

سناریو اصلی:

۱. شروع

۲. مهمان دکمه ورود را انتخاب می‌کند و سیستم فرم ورود را به مهمان نمایش می‌دهد.

۳. مهمان فرم ورود را تکمیل می‌کند و با دکمه ارسال، فرم تکمیل شده را به سیستم ارسال می‌کند.

۴. سیستم فرم ورود را بررسی می‌کند و اطلاعات را از بانک اطلاعات دریافت می‌کند.

۵. کاربر در سایت شناسایی و وارد می‌شود.

۶. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در اطلاعات فرم ورود

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در اطلاعات فرم ورود اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به مهمان نمایش داده می‌شود و درخواست اصلاح اطلاعات فرم را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: کاربر با موفقیت وارد می‌شود.
شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت آمیز بودن ورود اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و یک پیغام به کاربر نمایش داده می‌شود که ورود موفقیت آمیزی داشته.

۳. از مرحله ۵ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

سناریو فرعی ۳: کاربر وجود ندارد.
شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت وجود نداشتن اطلاعات در بانک اطلاعات اجرا می‌شود.

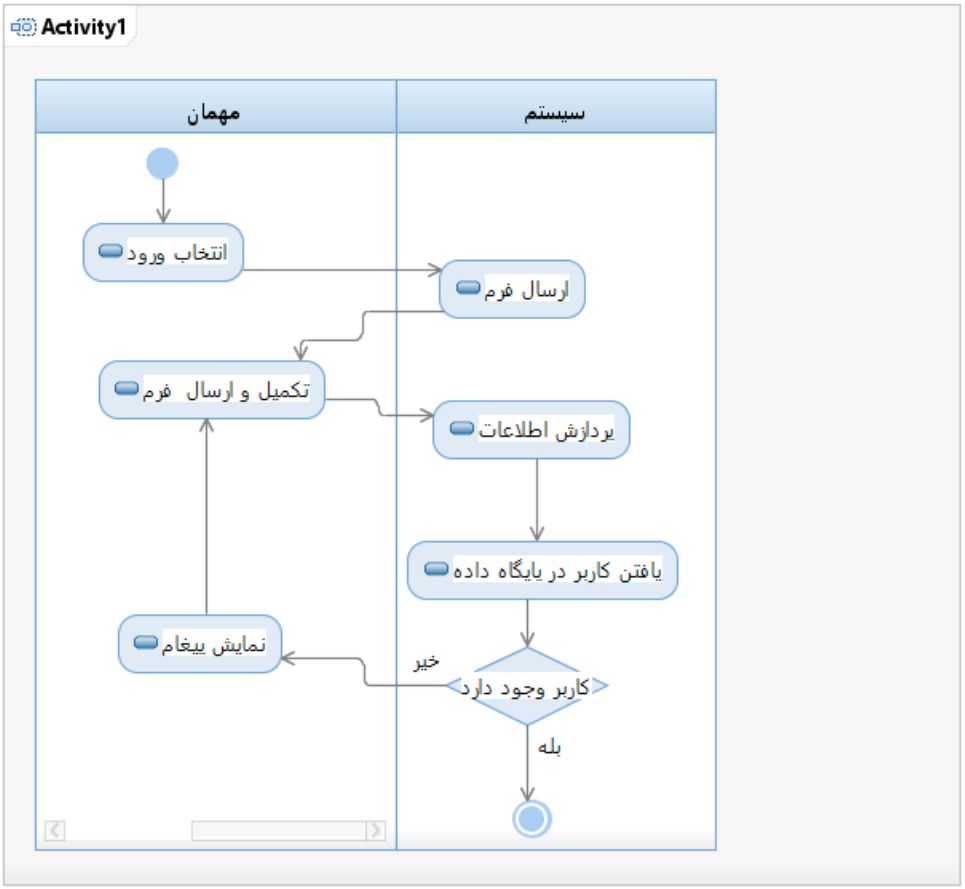
۱. شروع

۲. اطلاعات فرم بررسی می‌شود و وجود نداشتن کاربر مشخص می‌شوند.

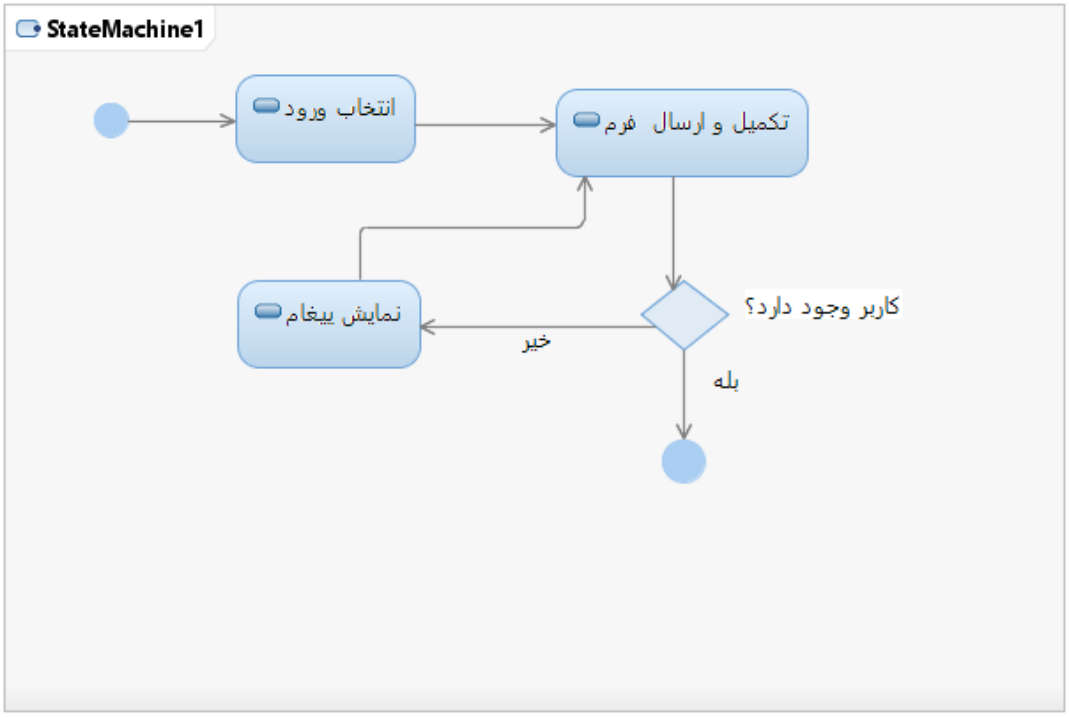
۳. یک پیغام به مهمان نمایش داده می‌شود و دکمه ثبت‌نام نمایش داده می‌شود.

۴. پایان

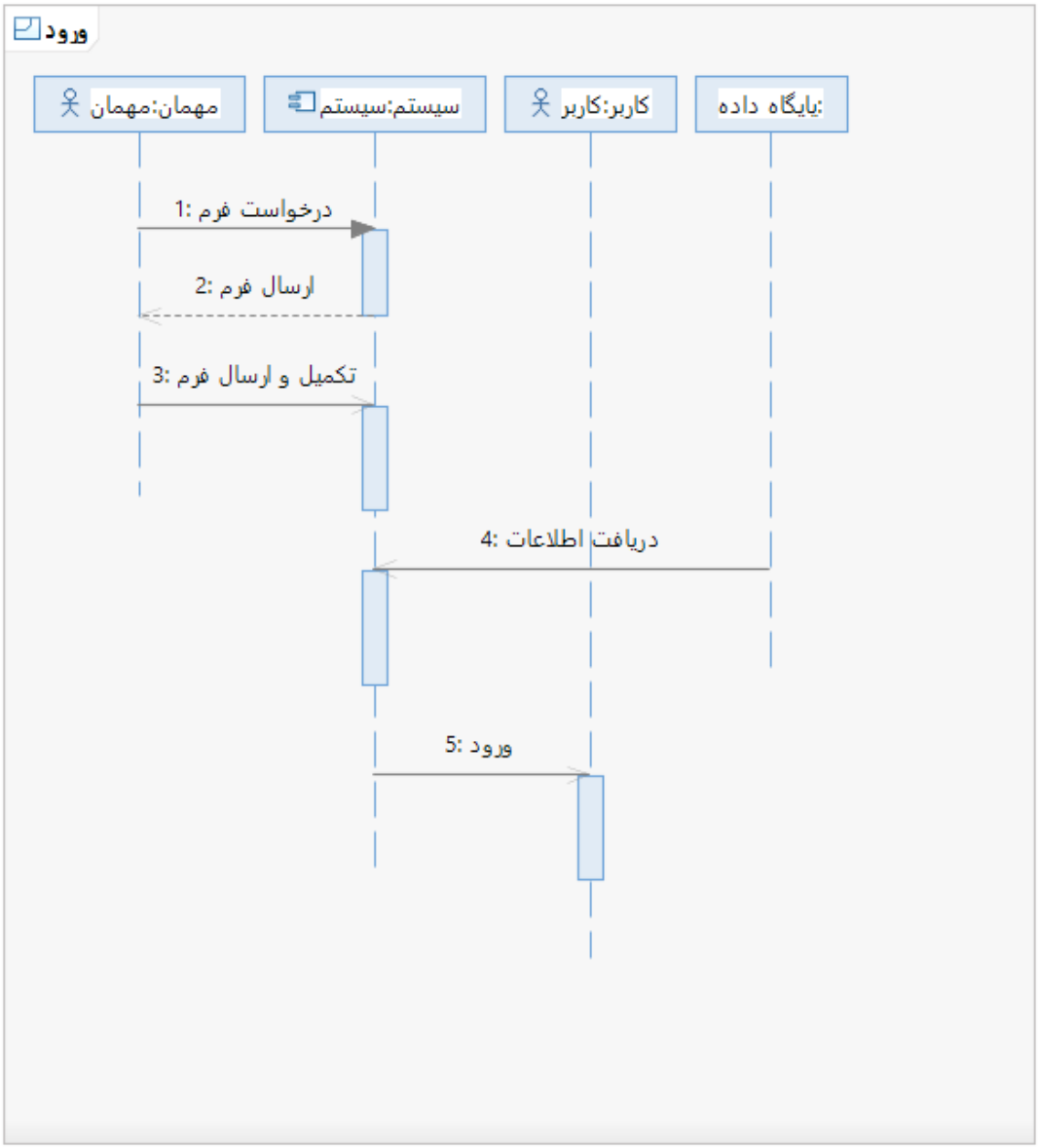
پس شرط: ندارد .



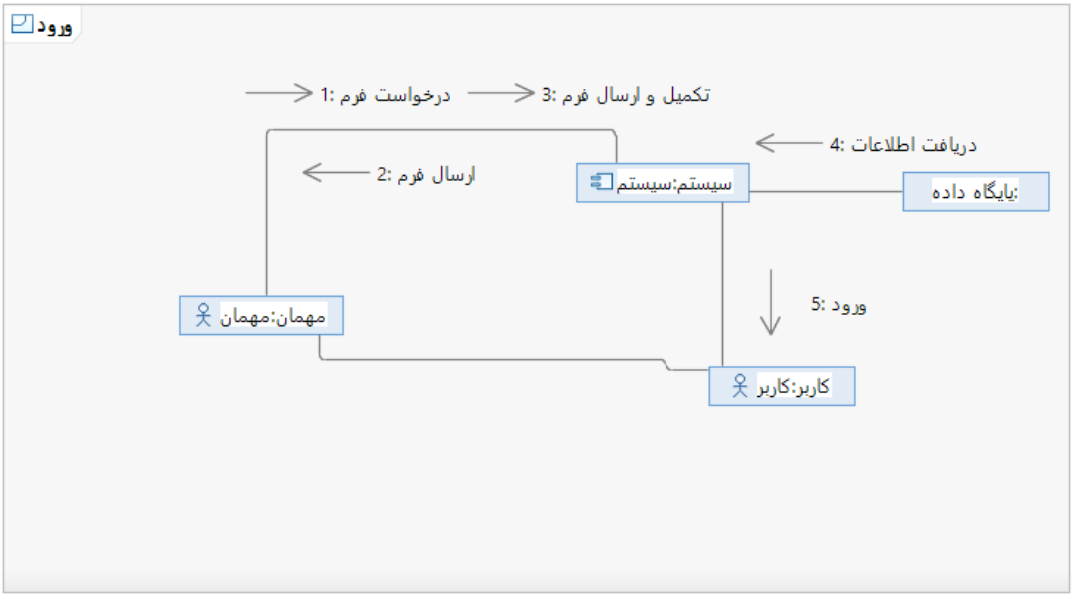
شکل ۶.۳: دیاگرام فعالیت ورود



شکل ۷.۳: دیاگرام حالت ماشین ورود



شکل ۸.۳: دیاگرام توالی ورود



شکل ۹.۳: دیاگرام همکار ورود

۳.۳ لیست پروژه‌ها

مورد استفاده: نمایش پروژه‌ها

شرح مختصر UC: نمایش پروژه‌های فعال در سایت.

پیش شرط: ندارد.

سناریو اصلی:

۱. شروع

۲. مهمان دکمه پروژه‌ها را انتخاب می‌کند و سیستم کل پروژه‌ها را به مهمان نمایش می‌دهد.

۳. مهمان با انتخاب هر پروژه به اطلاعات آن دسترسی پیدا می‌کند.

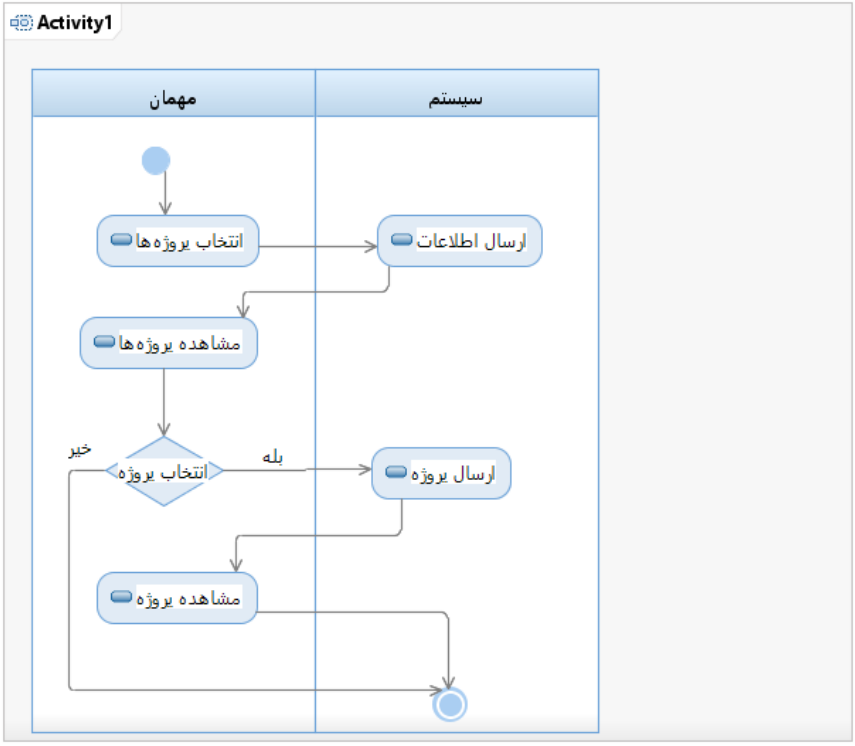
۴. پایان

پس شرط: ندارد.

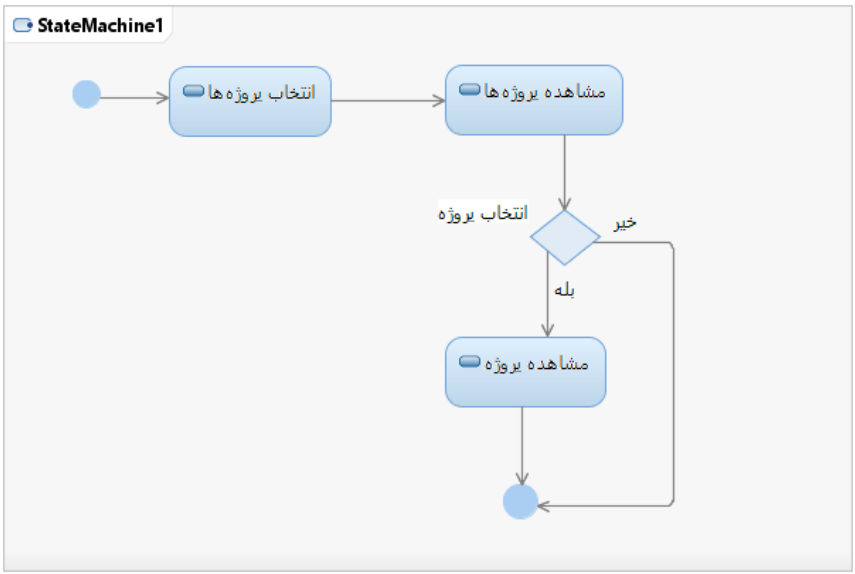
سناریوهای فرعی:

ندارد.

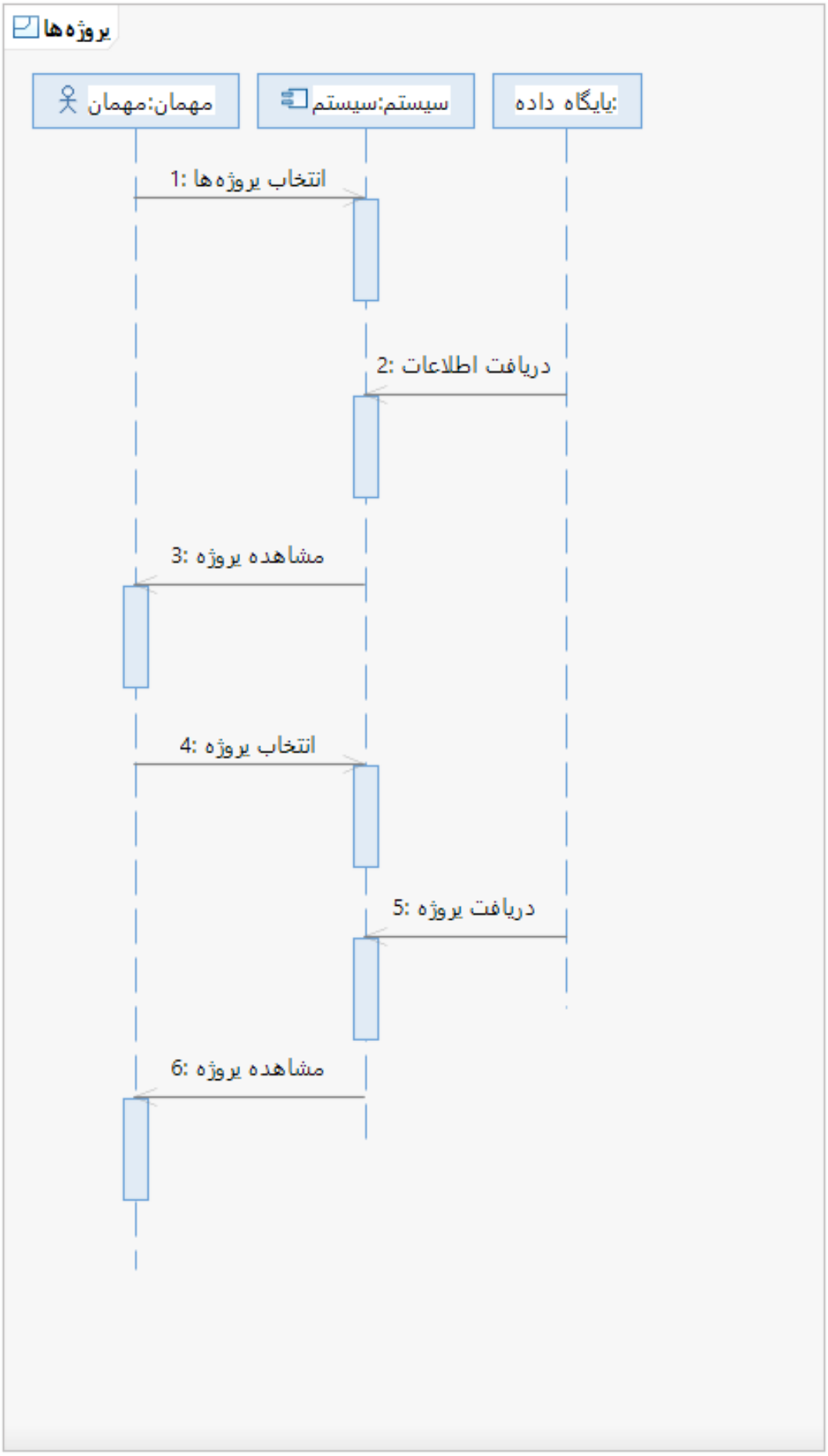
پس شرط: ندارد.



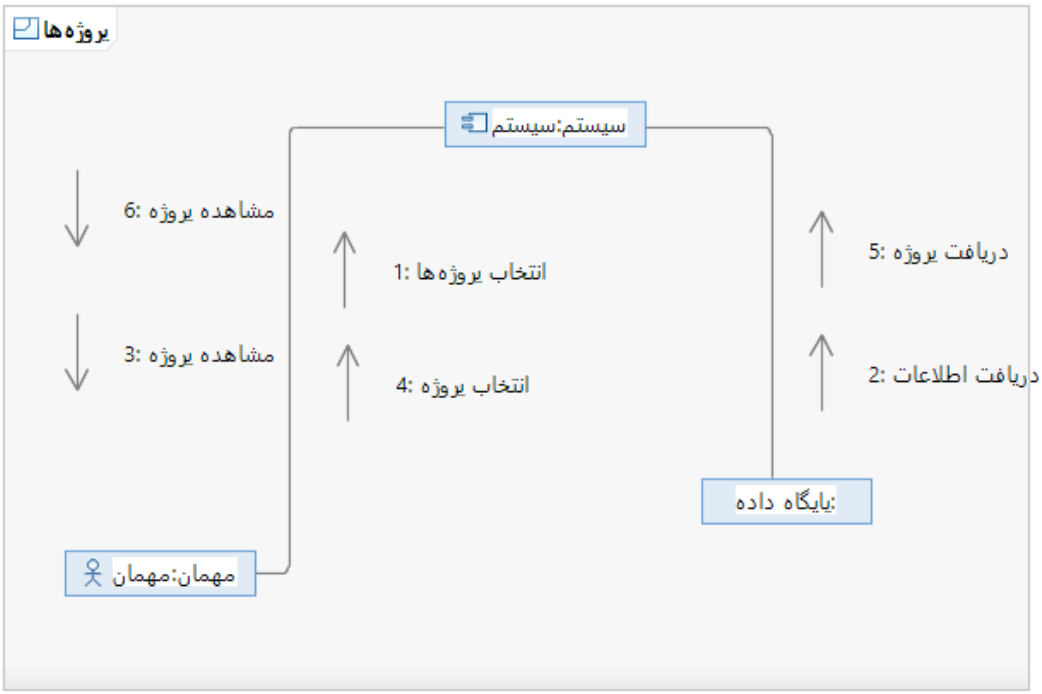
شکل ۱۰.۳: دیاگرام فعالیت لیست پروژه‌ها



شکل ۱۱.۳: دیاگرام حالت ماشین پروژه‌ها



شکل ۱۲.۳: دیاگرام توالی پروژه‌ها



شکل ۱۳.۳: دیاگرام همکار پروژه‌ها

۴.۳ لیست فریلنسرها

مورد استفاده: نمایش فریلنسرها

شرح مختصر UC: نمایش فریلنسرهای فعال در سایت.

پیش شرط: ندارد.

سناریو اصلی:

۱. شروع

۲. مهمان دکمه فریلنسرها را انتخاب می‌کند و سیستم فریلنسرها را به مهمان نمایش می‌دهد.

۳. مهمان با انتخاب هر فریلنسر به رزومه آن دسترسی پیدا می‌کند.

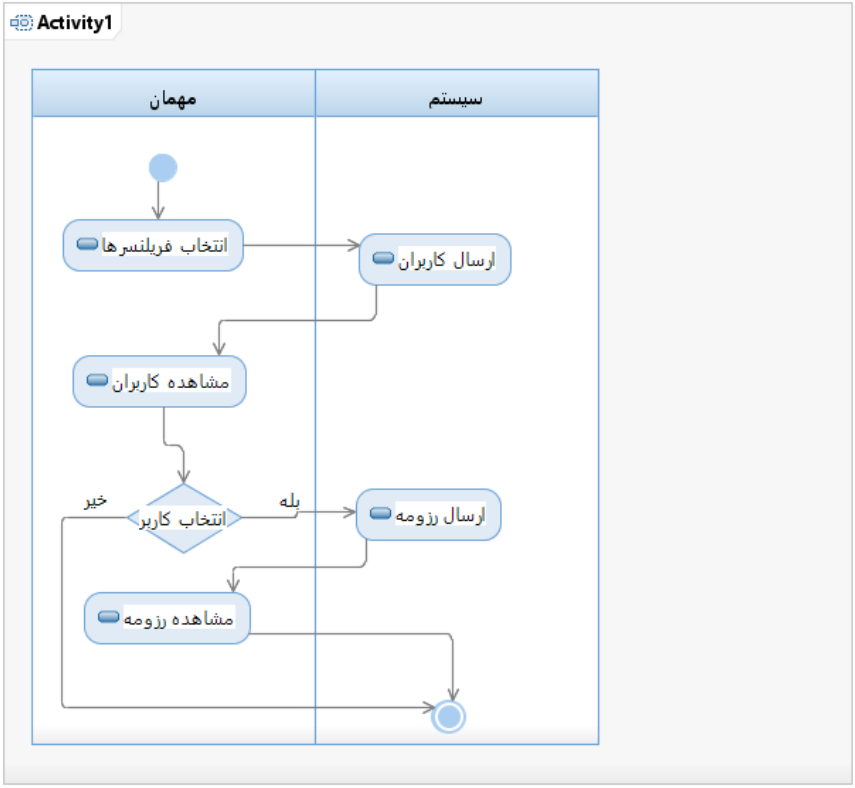
۴. پایان

پس شرط: ندارد.

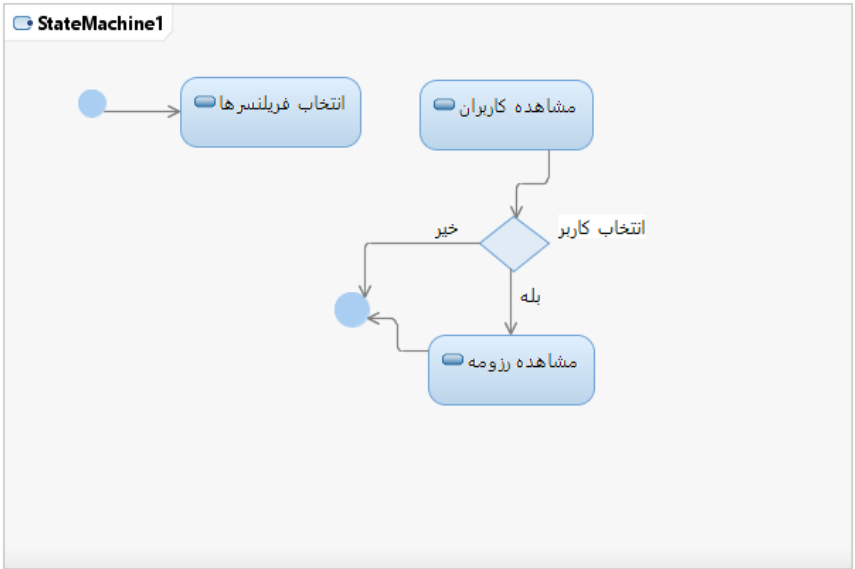
سناریوهای فرعی:

ندارد.

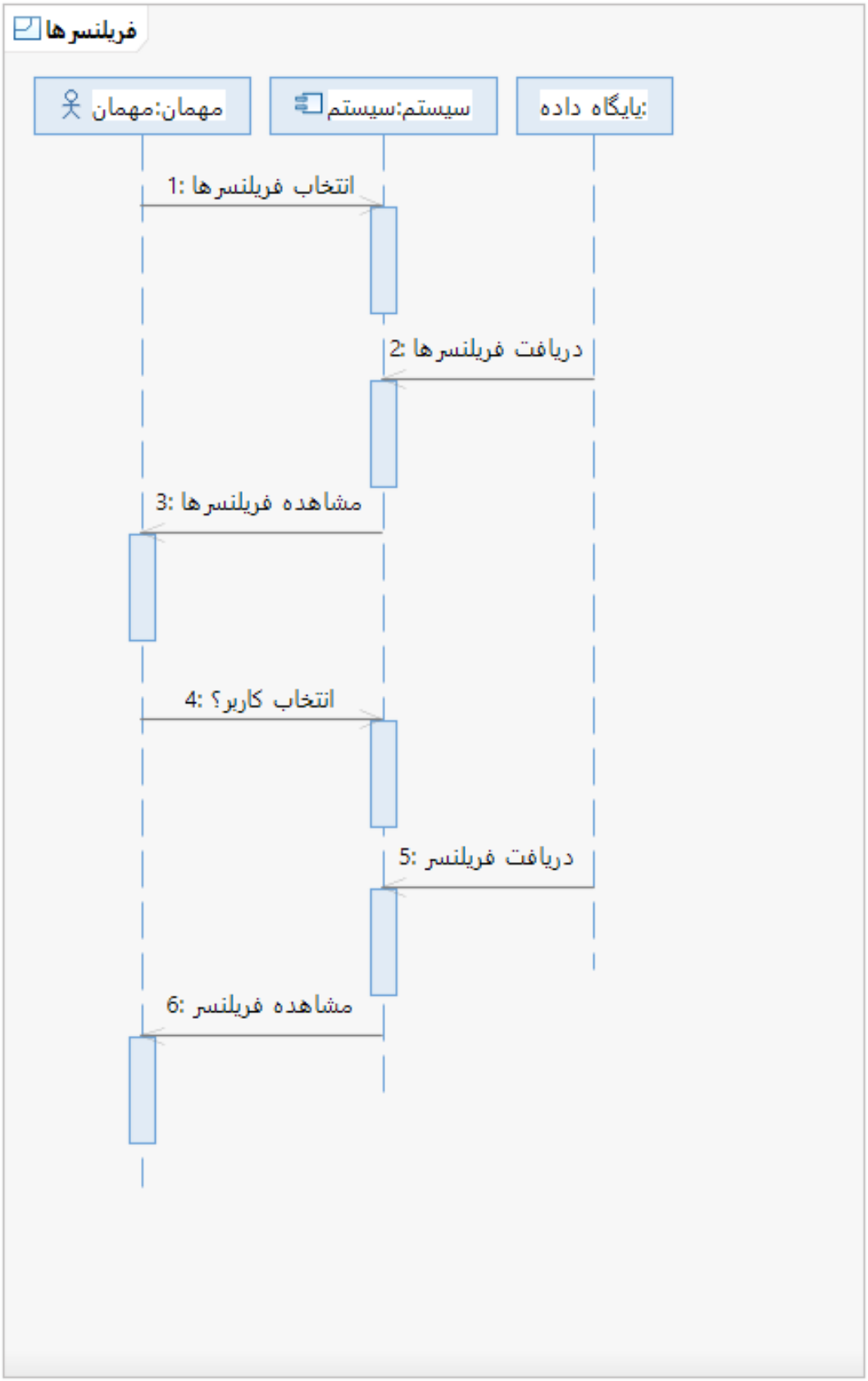
پس شرط: ندارد.



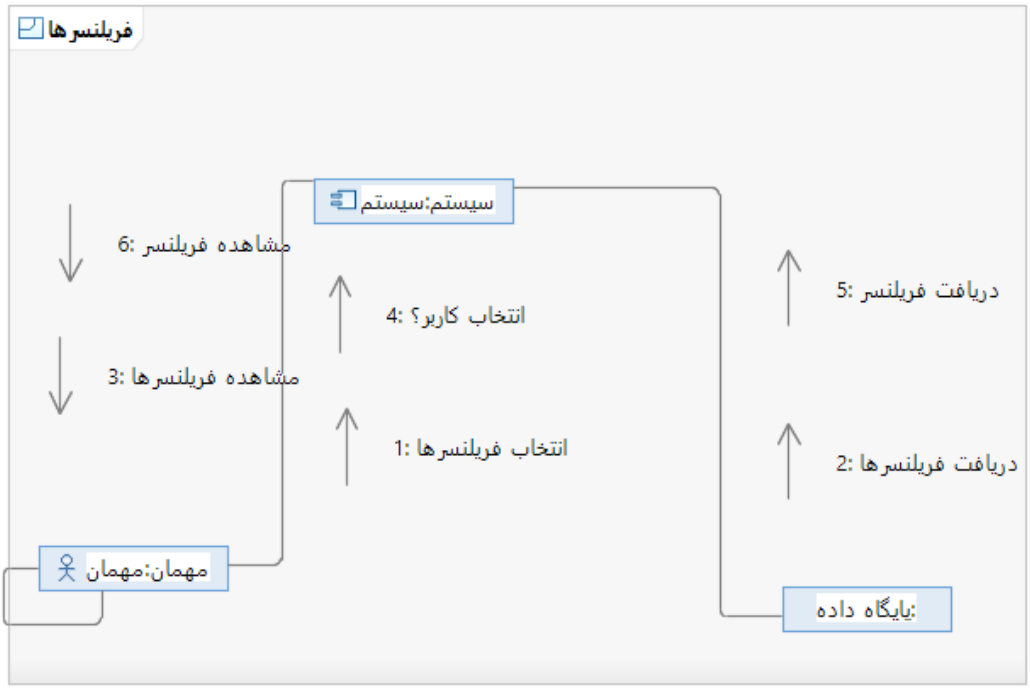
شکل ۱۴.۳: دیاگرام فعالیت فریلنسرها



شکل ۱۵.۳: دیاگرام حالت ماشین فریلنسرها



شکل ۱۶.۳: دیاگرام توالی فریلنسرها



شکل ۱۷.۳: دیاگرام همکار فریلنسرها

۵.۳ داشبورد کاربر

مورد استفاده: داشبورد کاربر

شرح مختصر UC: در این قسمت دو داشبورد فریلنسر و کارفرما را در اختیار کاربر قرار می‌دهد.

پیش شرط: ورود به سایت.

سناریو اصلی:

۱. شروع

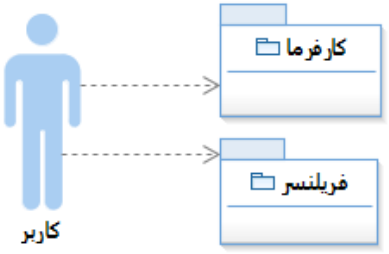
۲. کاربر با انتخاب هر کدام از داشبوردها به عنوان کارفرما / فریلنسر به سیستم معرفی می‌شود.

۳. پایان

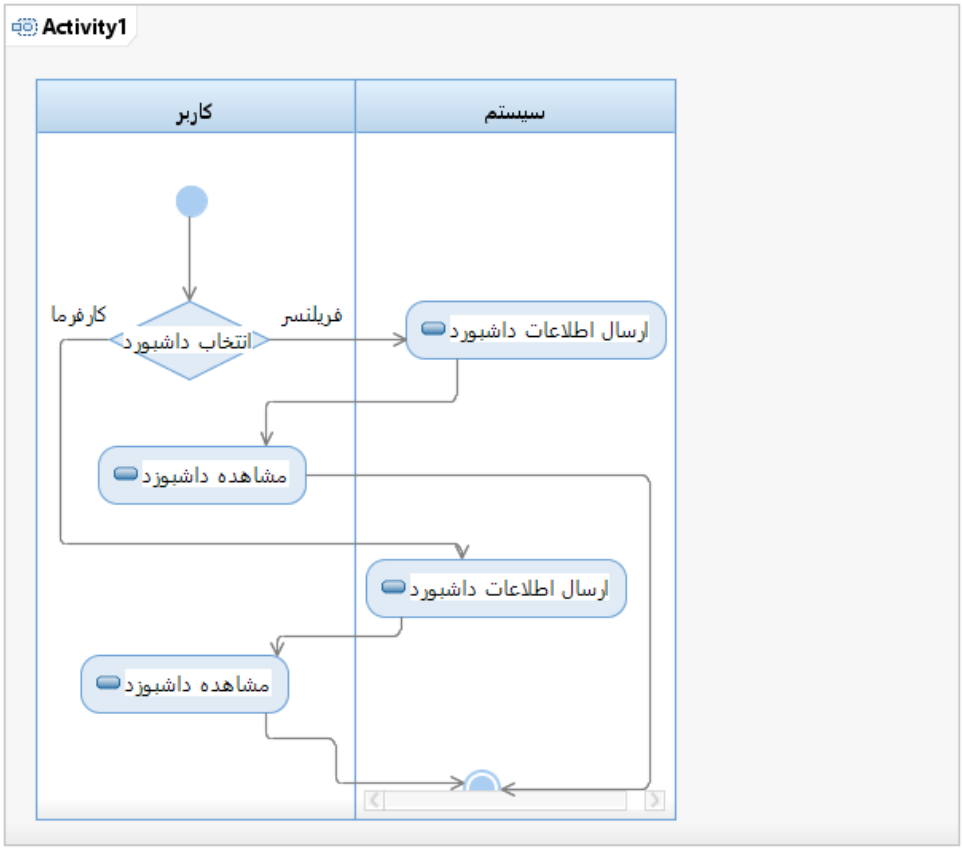
پس شرط: ندارد .

سناریوهای فرعی: ندارد.

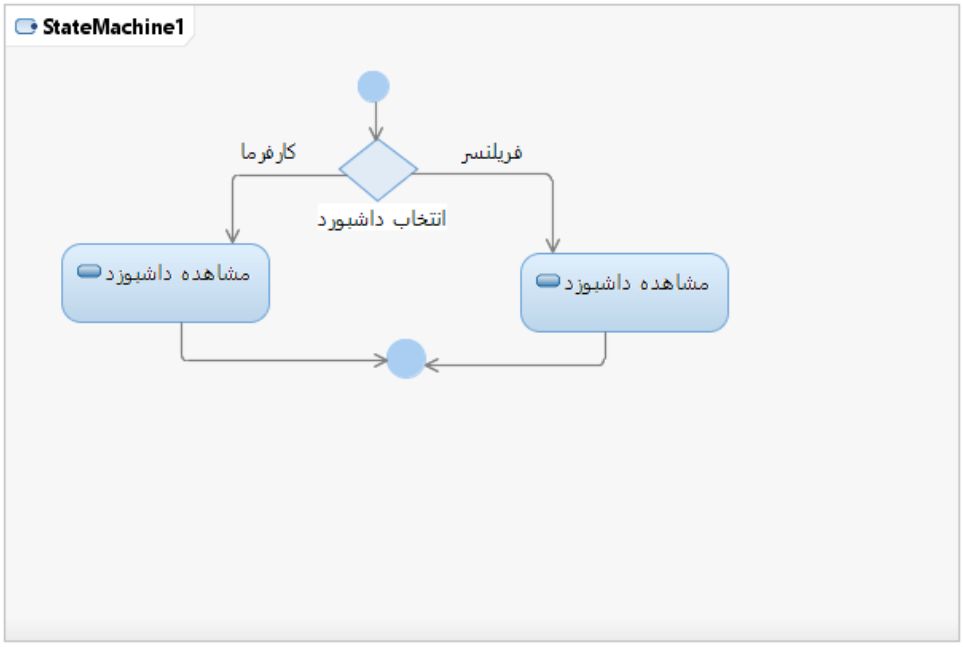
پس شرط: ندارد .



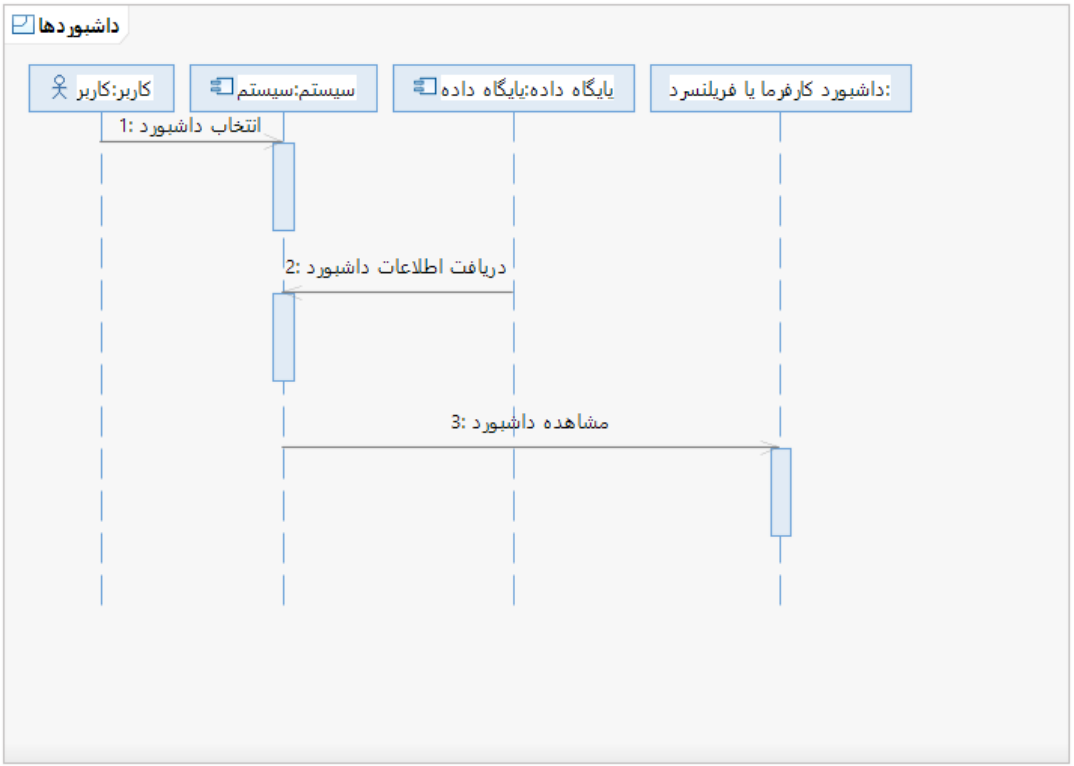
شکل ۱۸.۳: دیاگرام UC داشبورد کاربر



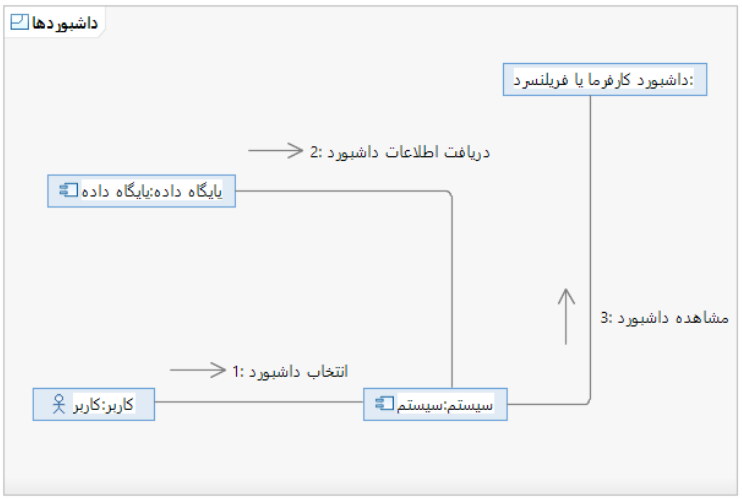
شکل ۱۹.۳: دیاگرام فعالیت داشبورد کاربر



شکل ۲۰.۳: دیاگرام حالت ماشین داشبورد کاربر



شکل ۲۱.۳: دیاگرام توالی داشبورد کاربر



شکل ۲۲.۳: دیاگرام همکار داشبورد کاربر

۶.۳ داشبورد کارفرما

مورد استفاده: داشبورد کارفرما

شرح مختصر UC: در این قسمت داشبورد کارفرما را در اختیار کاربر قرار می‌دهد.

پیش شرط: ورود به داشبورد کارفرما.

سناریو اصلی:

۱. شروع

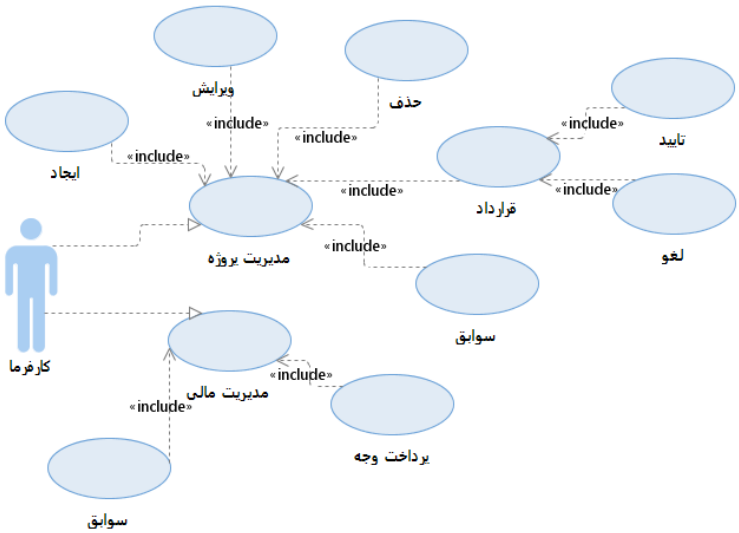
۲. کارفرما به بخش‌های مختلف مانند ایجاد و اصلاح پروژه، انتخاب فریلنسر برای پروژه و .. دسترسی پیدا می‌کند.

۳. پایان

پس شرط: ندارد.

سناریوهای فرعی: ندارد.

پس شرط: ندارد.



شکل ۲۳.۳: دیاگرام UC داشبورد کارفرما

۱.۶.۳ ایجاد پروژه

مورد استفاده: ایجاد پروژه

شرح مختصر UC: در این قسمت کارفرما پروژه خود را تعریف می‌کند.

پیش شرط: ورود به مدیریت پروژه در داشبورد کارفرما.

سناریو اصلی:

۱. شروع

۲. کارفرما دکمه ایجاد پروژه را انتخاب می‌کند و سیستم فرم خام را به کارفرما نمایش می‌دهد.

۳. کارفرما فرم را تکمیل می‌کند و با دکمه ارسال، فرم تکمیل شده را به سیستم ارسال می‌کند.

۴. سیستم اطلاعات فرم را بررسی می‌کند و اطلاعات را در بانک اطلاعات ثبت می‌کند.

۵. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در اطلاعات فرم ایجاد پروژه

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در اطلاعات فرم اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به کارفرما نمایش داده می‌شود و درخواست اصلاح اطلاعات فرم را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: با موفقیت ثبت شود

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت‌آمیز بودن ایجاد پروژه اجرا می‌شود.

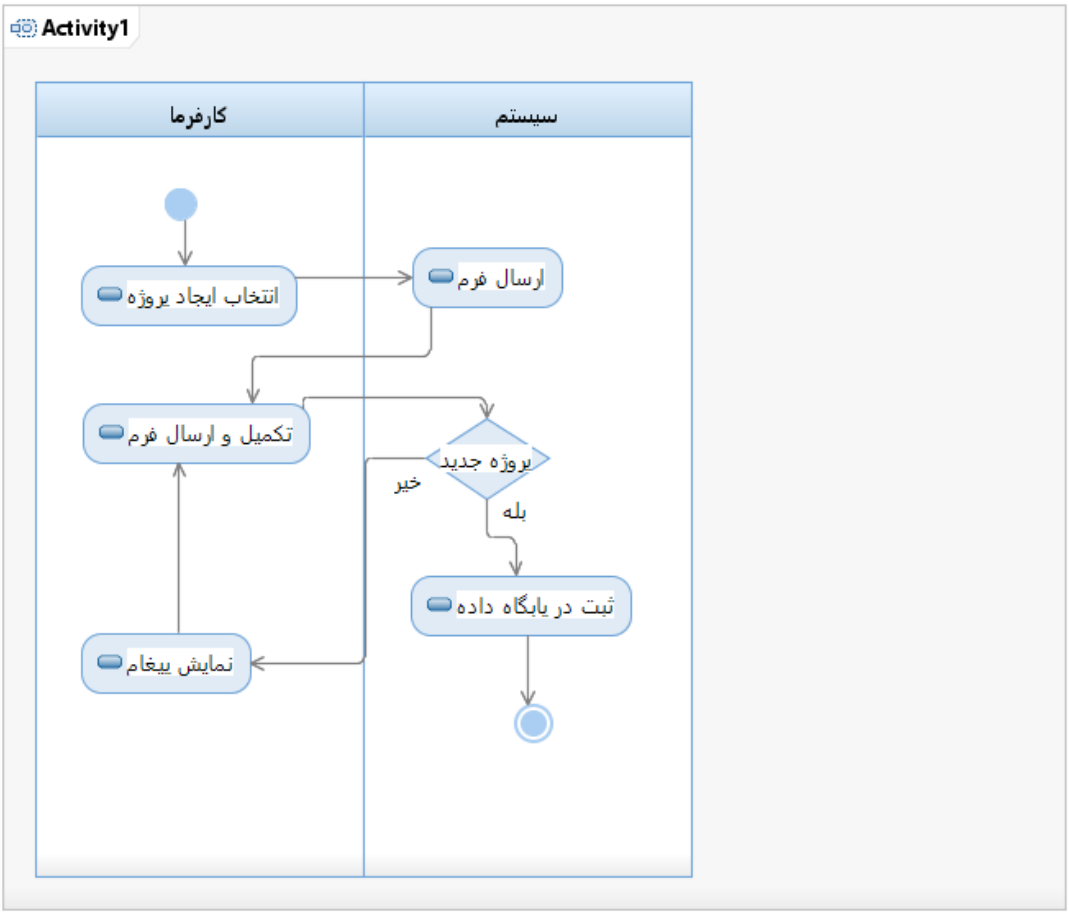
۱. شروع

۲. اطلاعات فرم بررسی می‌شود و یک پیغام به کارفرما نمایش داده می‌شود که اطلاعات با موفقیت ثبت شده است.

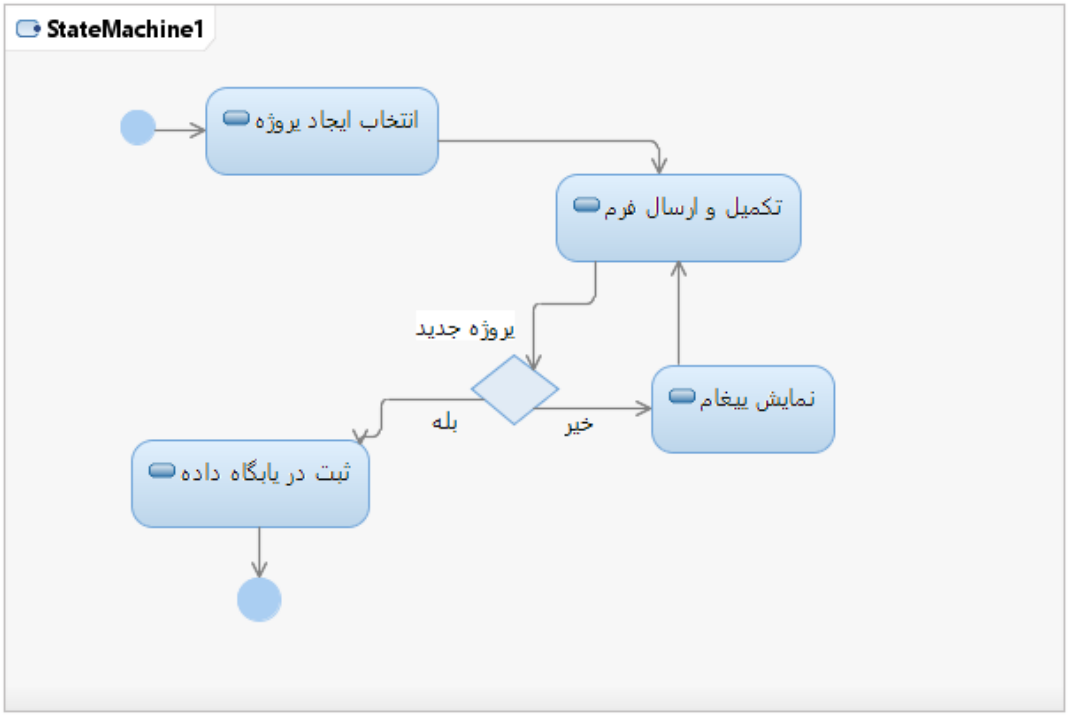
۳. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

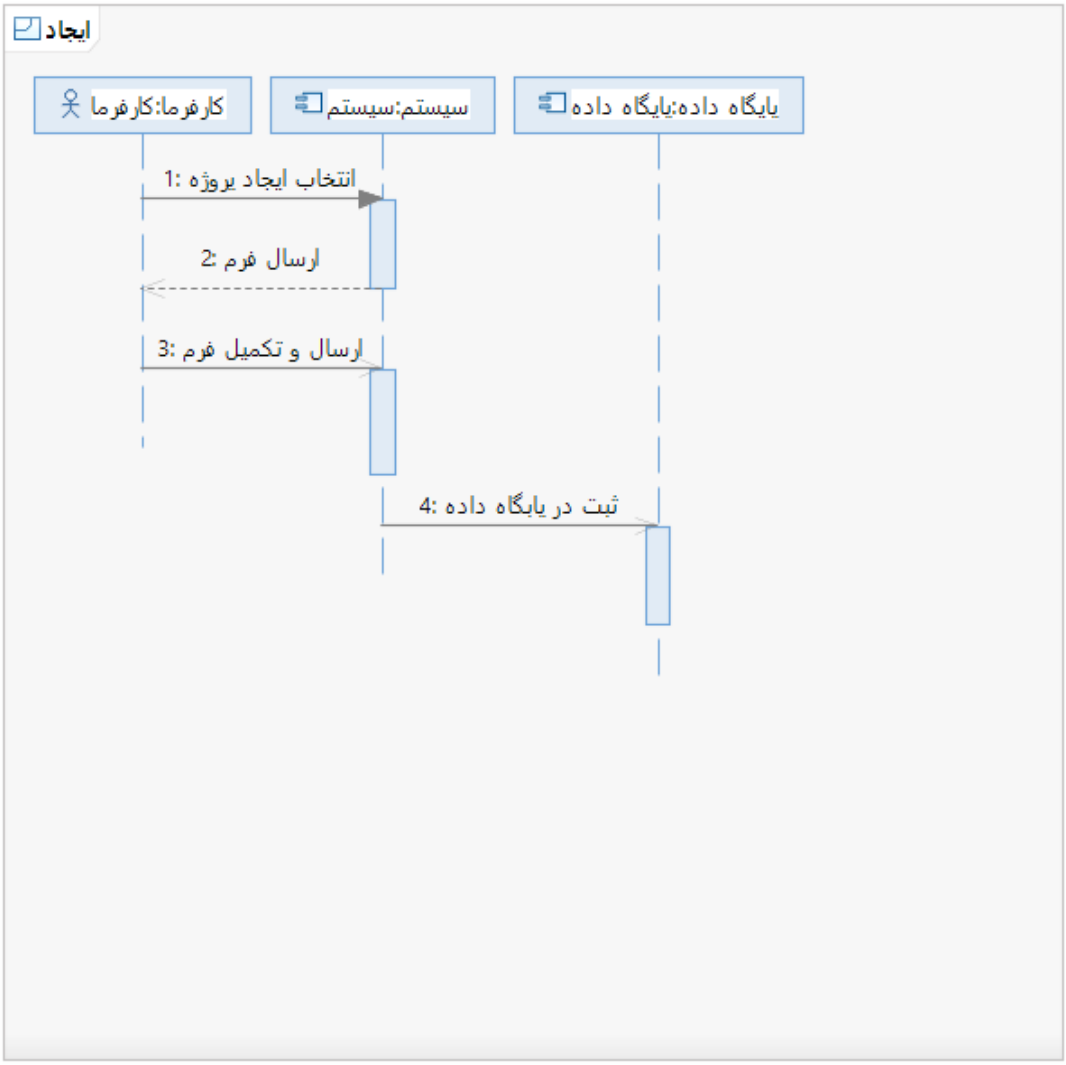
پس شرط: ندارد.



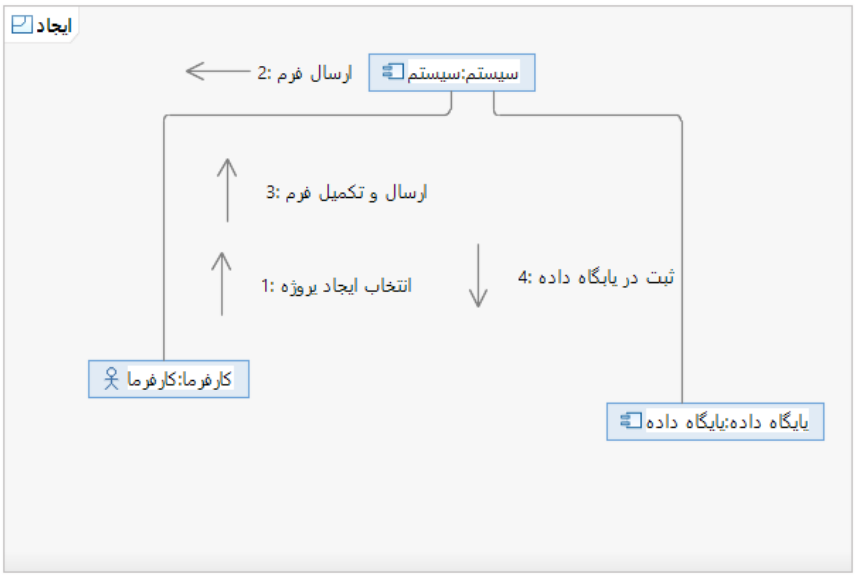
شکل ۲۴.۳: دیاگرام فعالیت ایجاد پروژه



شکل ۲۵.۳: دیاگرام حالت ماشین ایجاد پروژه



شکل ۲۶.۳: دیاگرام توالی ایجاد پروژه



شکل ۲۷.۳: دیاگرام همکار ایجاد پروژه

۲.۶.۳ ویرایش پروژه

مورد استفاده: ویرایش پروژه

شرح مختصر UC: در این قسمت کارفرما پروژه خود را اصلاح می‌کند.

پیش شرط: ورود به مدیریت پروژه در داشبورد کارفرما.

سناریو اصلی:

۱. شروع

۲. کارفرما دکمه ویرایش پروژه را انتخاب می‌کند و سیستم فرم اطلاعات پروژه را به کارفرما نمایش می‌دهد.

۳. کارفرما فرم را اصلاح می‌کند و با دکمه ارسال، فرم اصلاح شده را به سیستم ارسال می‌کند.

۴. سیستم اطلاعات فرم را بررسی می‌کند و اطلاعات را در بانک اطلاعات بروزرسانی می‌کند.

۵. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در اطلاعات فرم ویرایش پروژه

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در اطلاعات فرم اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به کارفرما نمایش داده می‌شود و درخواست اصلاح اطلاعات فرم را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: اطلاعات با موفقیت اصلاح شود.

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت آمیز بودن اصلاح اطلاعات پروژه اجرا می‌شود.

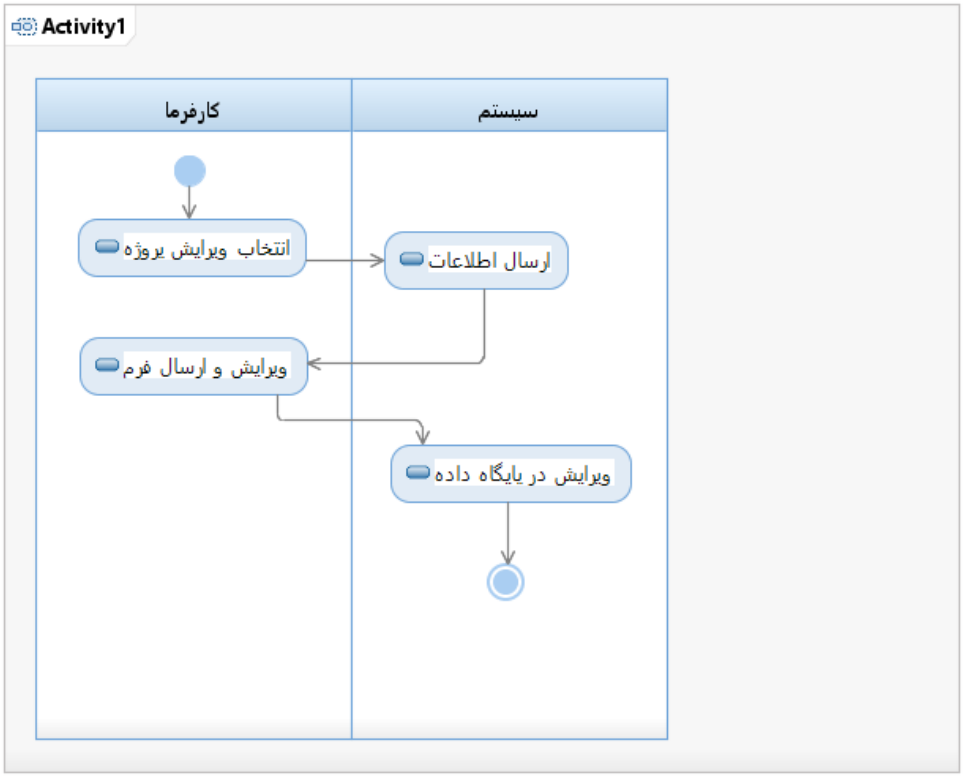
۱. شروع

۲. اطلاعات فرم بررسی می‌شود و یک پیغام به کارفرما نمایش داده می‌شود که اطلاعات با موفقیت ثبت شده است.

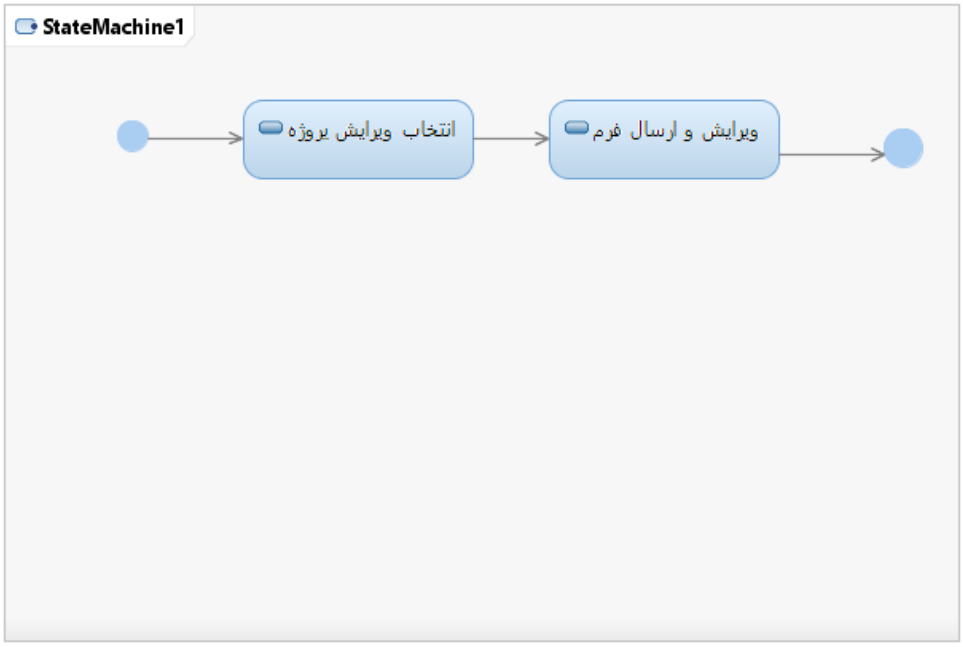
۳. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

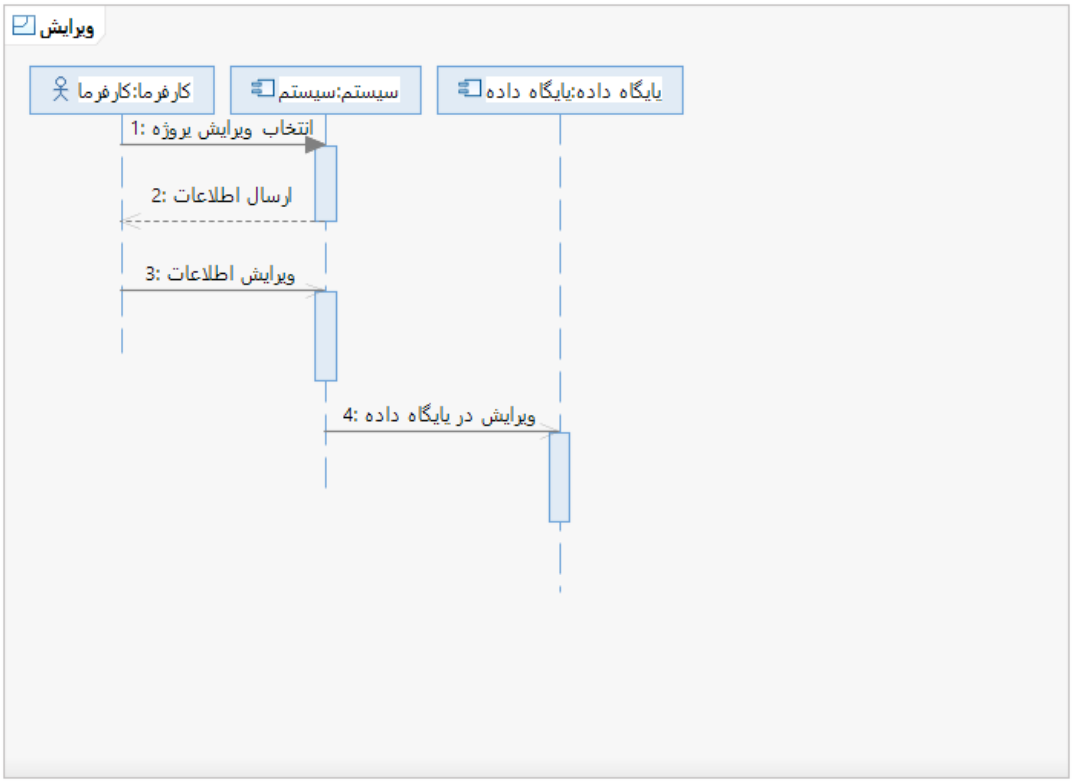
پس شرط: ندارد.



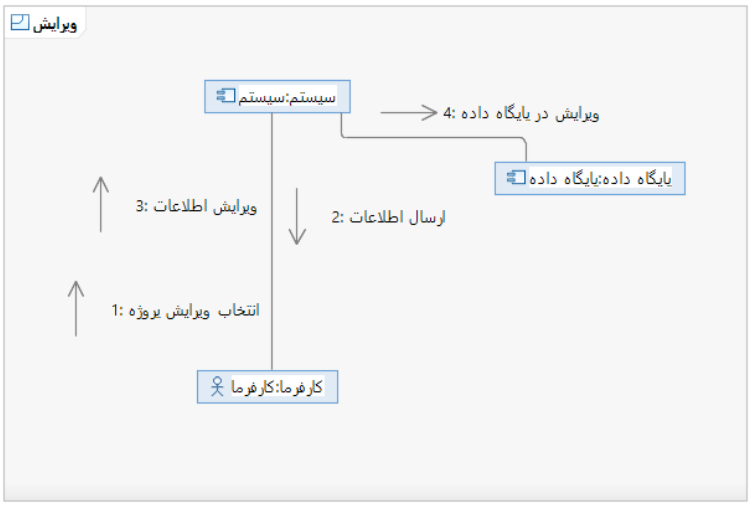
شکل ۲۸.۳: دیاگرام فعالیت ویرایش پروژه



شکل ۲۹.۳: دیاگرام حالت ماشین ویرایش پروژه



شکل ۳۰.۳: دیاگرام توالی ویرایش پروژه



شکل ۳۱.۳: دیاگرام همکار ویرایش پروژه

۳.۶.۳ حذف پروژه

مورد استفاده: حذف پروژه

شرح مختصر UC: در این قسمت کارفرما پروژه خود را حذف می‌کند.

پیش شرط: ورود به مدیریت پروژه در داشبورد کارفرما.

سناریو اصلی:

۱. شروع

۲. کارفرما دکمه حذف پروژه را انتخاب می‌کند و سیستم اطلاعات را به کارفرما نمایش می‌دهد.

۳. کارفرما تایید حذف پروژه را به سیستم ارسال می‌کند.

۴. سیستم پروژه را بررسی و از بانک اطلاعات حذف می‌کند.

۵. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در حذف پروژه

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در روند حذف پروژه اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به کارفرما نمایش داده می‌شود و درخواست اصلاح روند را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: با موفقیت حذف شود

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت‌آمیز بودن حذف پروژه اجرا می‌شود.

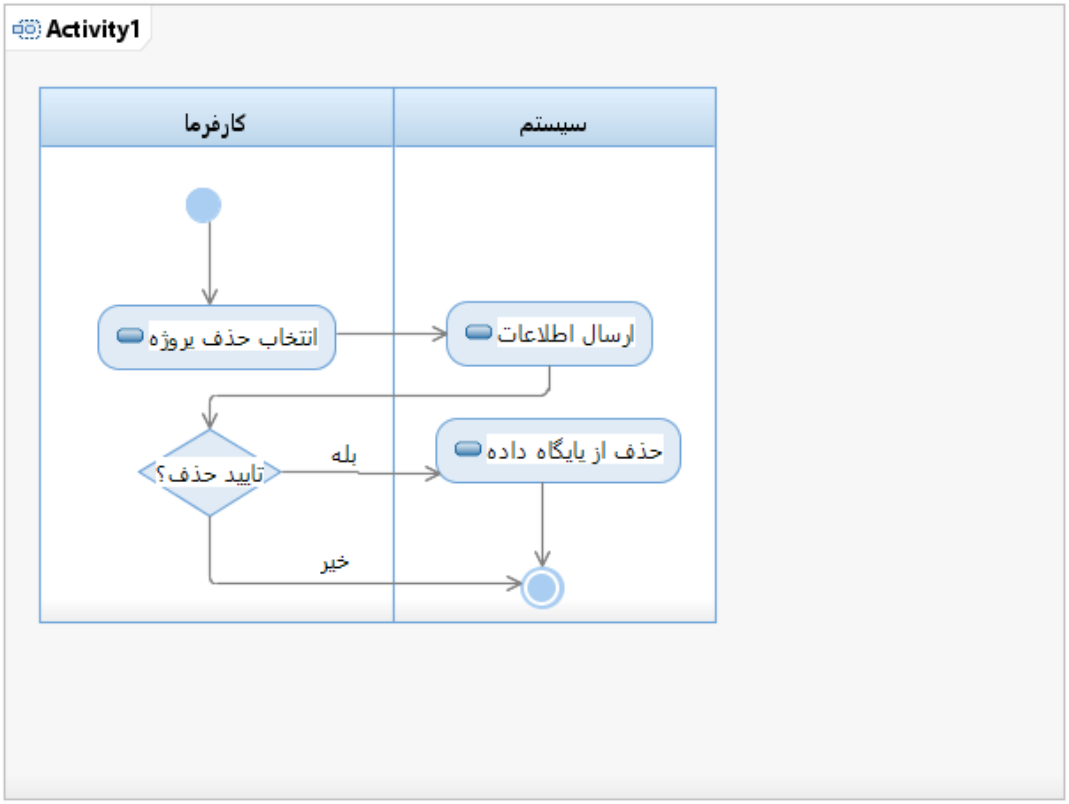
۱. شروع

۲. پروژه بررسی می‌شود و یک پیغام به کارفرما نمایش داده می‌شود که اطلاعات با موفقیت حذف شده است.

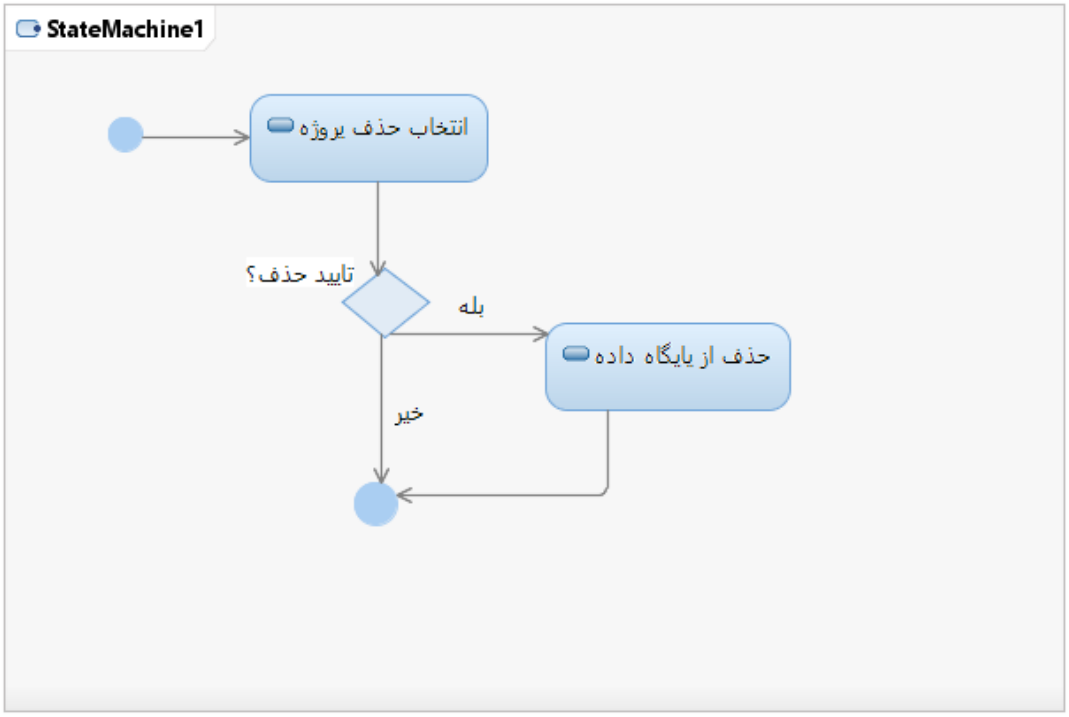
۳. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

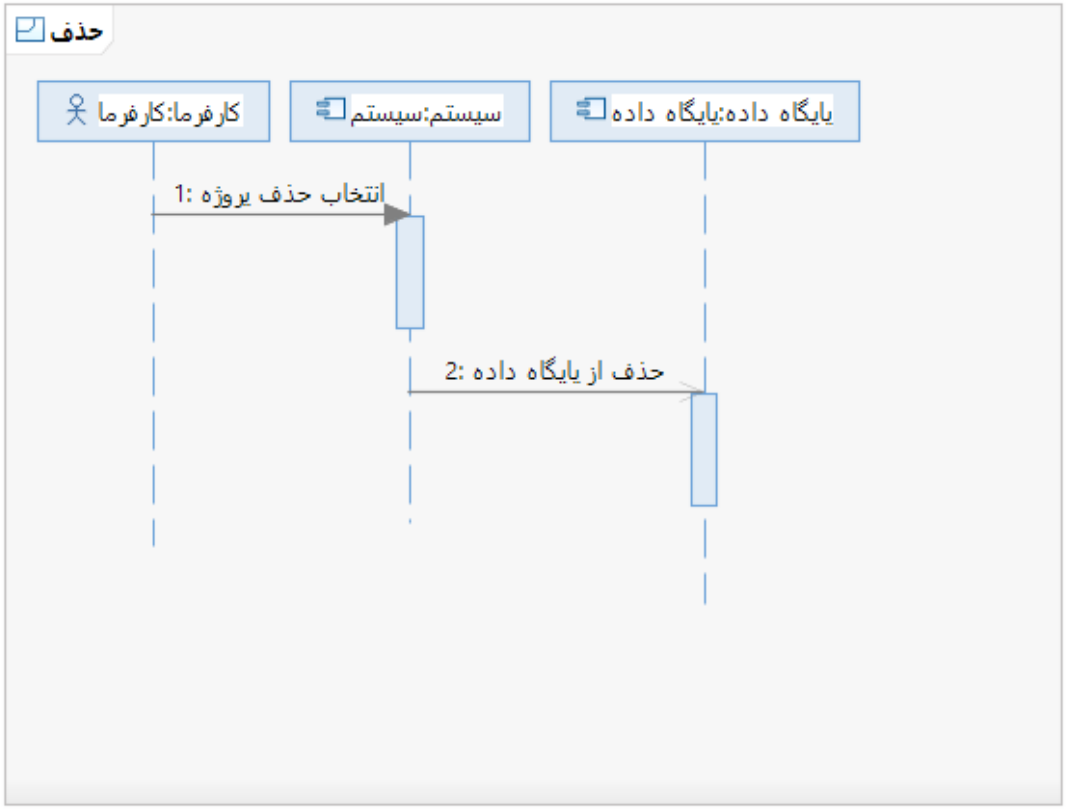
پس شرط: ندارد.



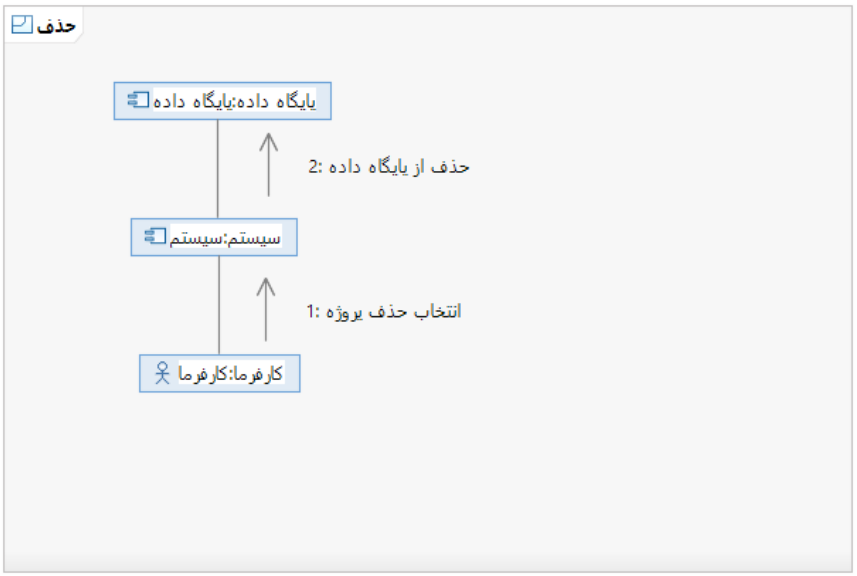
شکل ۳۲.۳: دیاگرام فعالیت حذف پروژه



شکل ۳۳.۳: دیاگرام حالت ماشین حذف پروژه



شکل ۳۴.۳: دیاگرام توالی حذف پروژه



شکل ۳۵.۳: دیاگرام همکار حذف پروژه

۴.۶.۳ پرداخت هزینه پروژه

مورد استفاده: پرداخت هزینه پروژه

شرح مختصر UC: در این قسمت کارفرما هزینه انجام پروژه توسط فریلنسر را پرداخت می‌کند.

پیش شرط: ورود به مدیریت مالی در داشبورد کارفرما.

سناریو اصلی:

۱. شروع

۲. کارفرما بعد از انتخاب پروژه، دکمه پرداخت هزینه پروژه را انتخاب می‌کند و سیستم فرم پرداخت را به کارفرما نمایش می‌دهد.

۳. کارفرما فرم را تکمیل می‌کند و با دکمه ارسال، فرم تکمیل شده را به سیستم ارسال می‌کند.

۴. سیستم اطلاعات فرم را بررسی می‌کند و اطلاعات را به بانک عامل ارسال می‌کند.

۵. اطلاعات پرداخت دریافت می‌شود و در بانک اطلاعات ثبت می‌شود.

۶. پایان

پس شرط: وجه بعد از انتخاب فریلنسر جهت ضمانت در سایت بلوکه می‌شود.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در اطلاعات فرم پرداخت

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در اطلاعات فرم اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به کارفرما نمایش داده می‌شود و درخواست اصلاح اطلاعات فرم را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: خطا در پرداخت وجه

شرح مختصر UC: این سناریو در مرحله ۵ سناریو اصلی در صورت خطا در پرداخت اجرا می‌شود.

۱. شروع

۲. اطلاعات خطا از طرف بانک عامل به سیستم ارسال می‌شود.

۳. یک پیغام به کارفرما نمایش داده می‌شود و ناموفق بودن پرداخت را اعلام می‌کند.

۴. از مرحله ۲ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۳: با موفقیت پرداخت شود

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت آمیز بودن پرداخت اجرا می‌شود.

۱. شروع

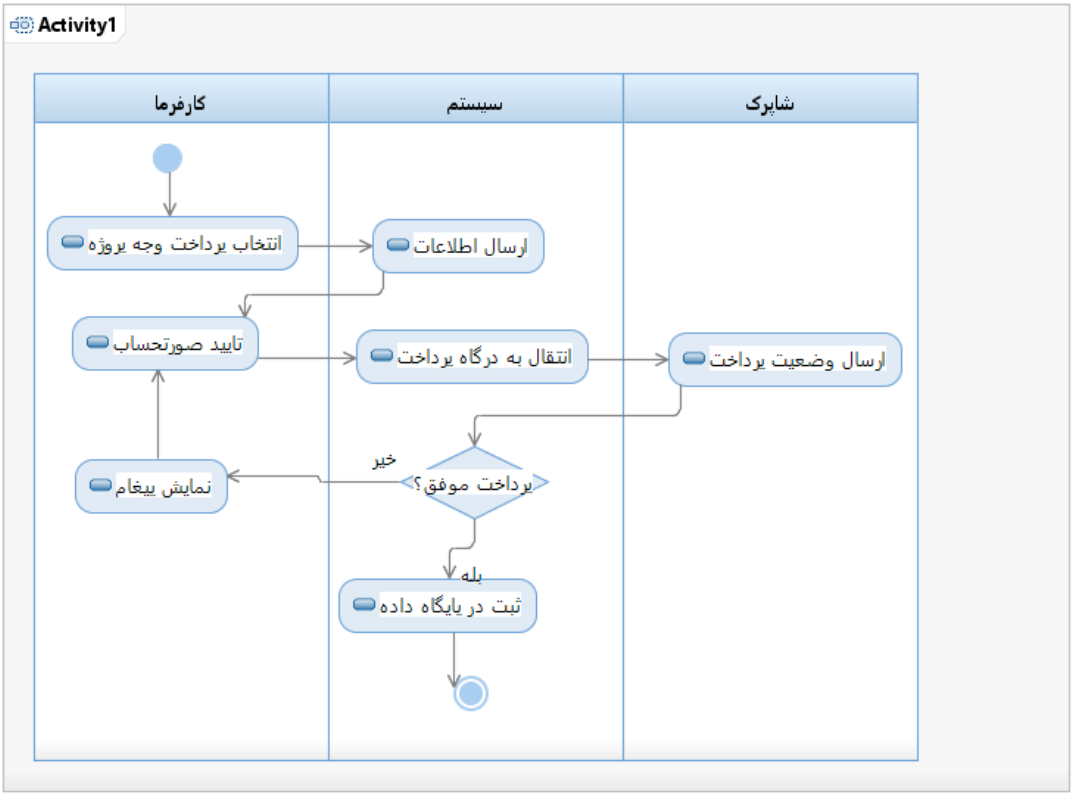
۲. اطلاعات پرداخت از طرف بانک عامل به سیستم ارسال می‌شود.

۳. یک پیغام به کارفرما نمایش داده می‌شود که پرداخت با موفقیت انجام شده است.

۴. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

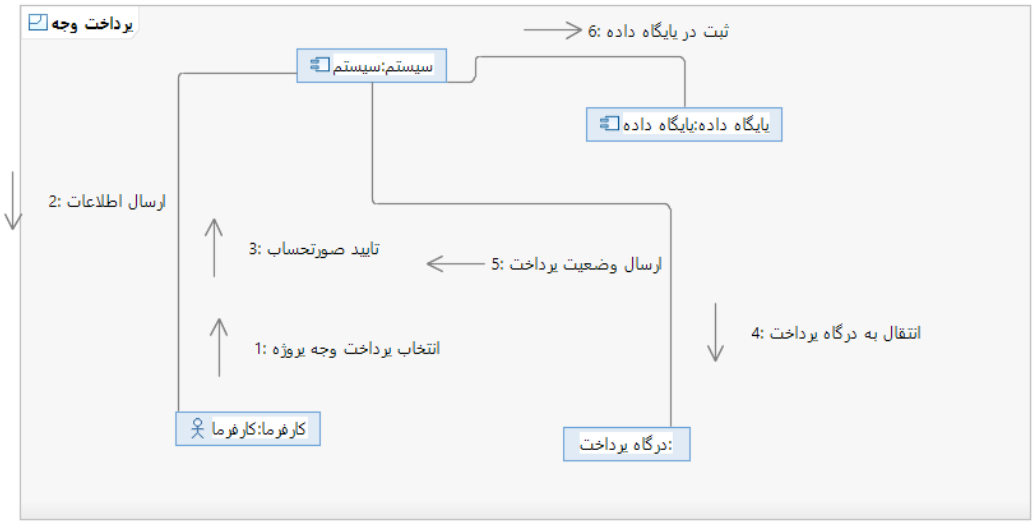
پس شرط: ندارد.



شکل ۳۶.۳: دیاگرام فعالیت پرداخت هزینه

شکل ۳۷.۳: دیاگرام حالت ماشین پرداخت هزینه

شکل ۳۸.۳: دیاگرام توالی پرداخت هزینه



شکل ۳۹.۳: دیاگرام همکار پرداخت هزینه

۵.۶.۳ واگذاری پروژه

مورد استفاده: واگذاری پروژه

شرح مختصر UC: در این قسمت کارفرما پروژه را به یک فریلنسر واگذار می‌کند.

پیش شرط: ورود به مدیریت پروژه در داشبورد کارفرما.

سناریو اصلی:

۱. شروع

۲. کارفرما دکمه قرارداد پروژه را انتخاب می‌کند و سیستم پیشنهادات فریلنسرها را به کارفرما نمایش می‌دهد.

۳. کارفرما بهترین پیشنهاد را انتخاب می‌کند و با دکمه تایید قرارداد را مشاهده کرده و وضعیت پیشنهاد را به سیستم ارسال می‌کند.

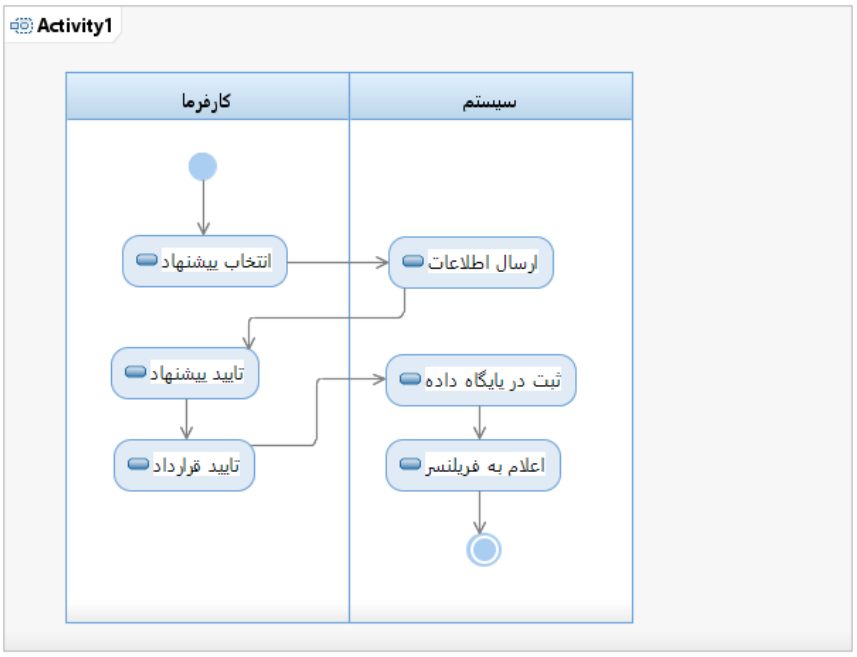
۴. سیستم به فریلنسر اطلاع داده و در بانک اطلاعات ثبت می‌کند.

۵. پایان

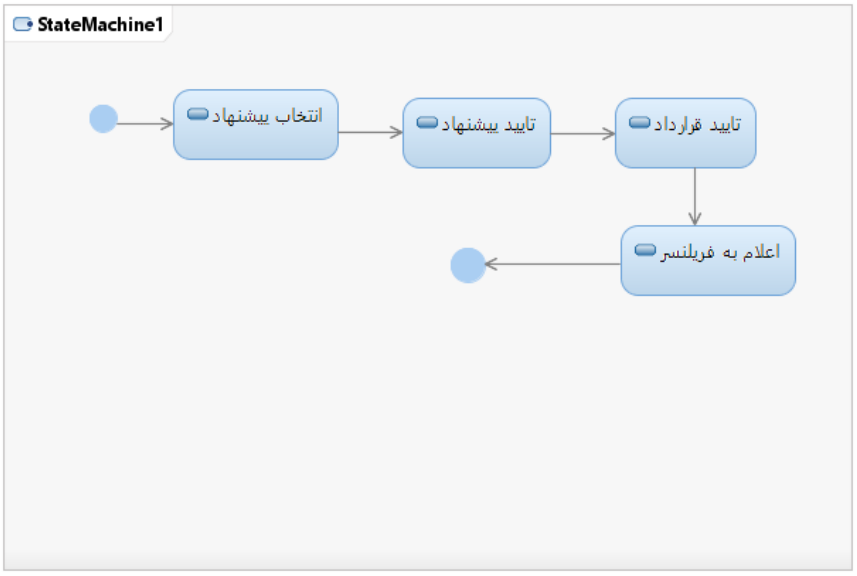
پس شرط: فریلنسر باید برای ادامه قرارداد را تایید کند.

سناریوهای فرعی: ندارد.

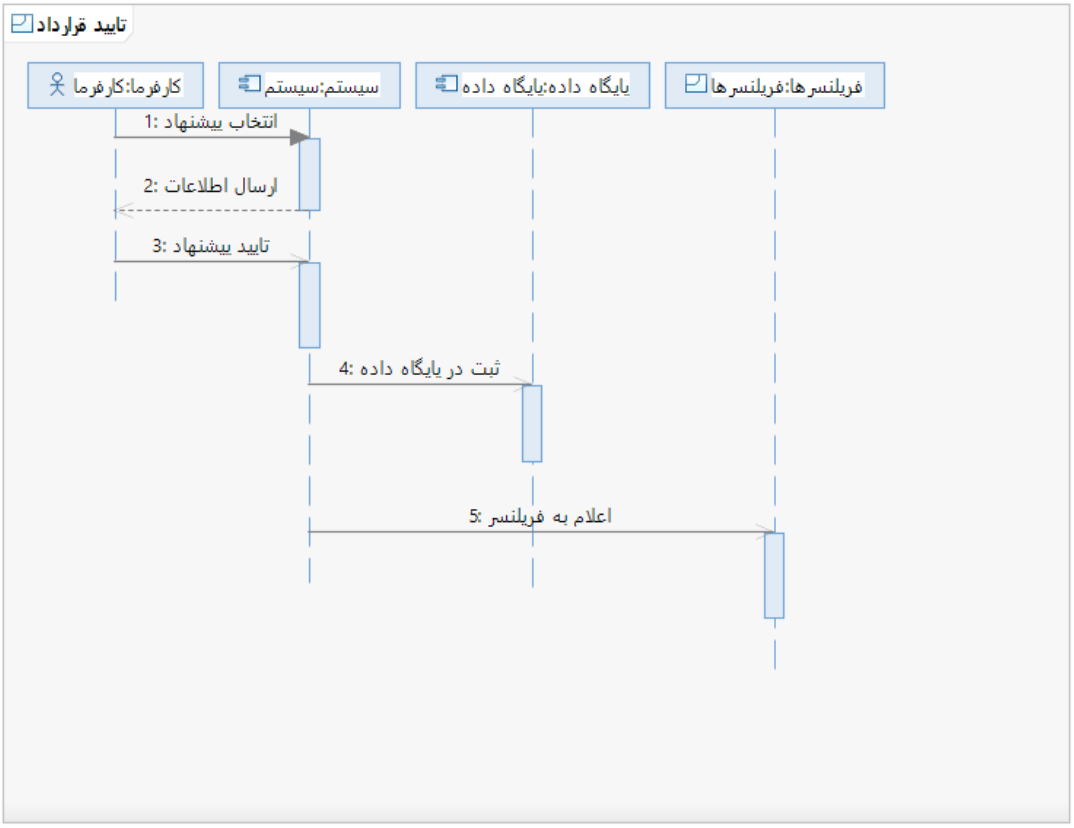
پس شرط: ندارد.



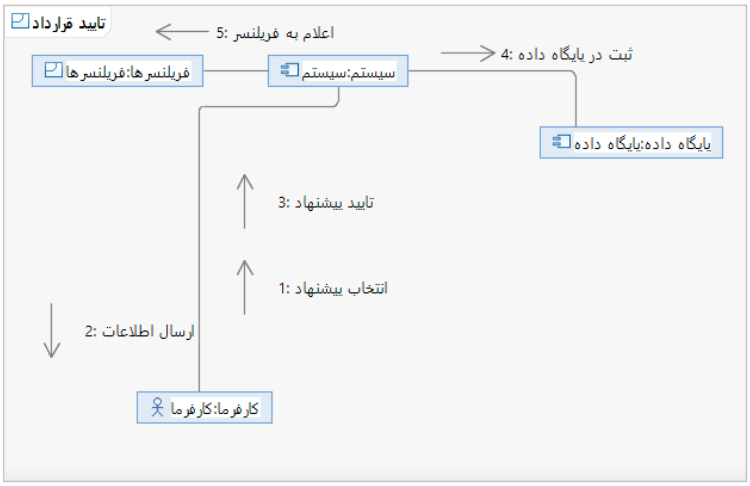
شکل ۴۰.۳: دیاگرام فعالیت واگذاری پروژه



شکل ۴۱.۳: دیاگرام حالت ماشین واگذاری پروژه



شکل ۴۲.۳: دیاگرام توالی واگذاری پروژه



شکل ۴۳.۳: دیاگرام همکار واگذاری پروژه

۷.۳ داشبورد فریلنسر

مورد استفاده: داشبورد فریلنسر

شرح مختصر UC: در این قسمت داشبورد فریلنسر را در اختیار کاربر قرار می‌دهد.

پیش شرط: ورود به داشبورد فریلنسر.

سناریو اصلی:

۱. شروع

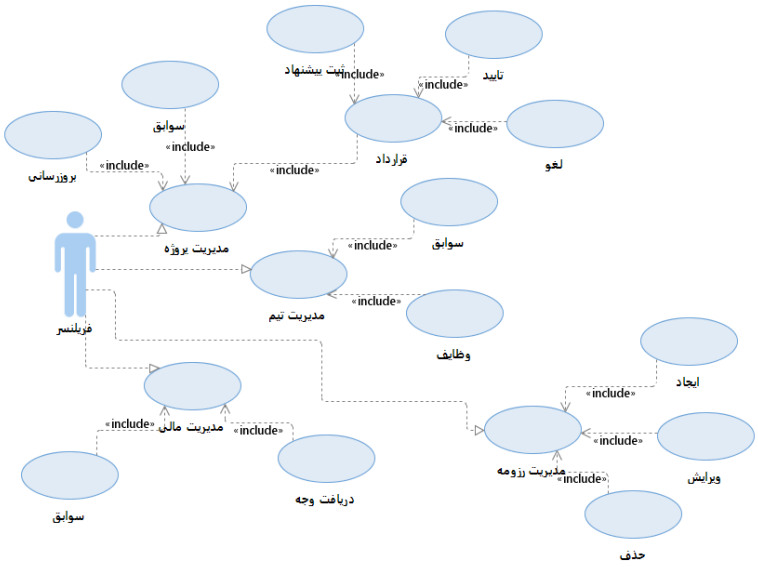
۲. فریلنسر به بخش‌های مختلف مانند ایجاد و اصلاح رزومه، پیشنهاد شرایط برای انجام پروژه کارفرما و .. دسترسی پیدا می‌کند.

۳. پایان

پس شرط: ندارد.

سناریوهای فرعی: ندارد.

پس شرط: ندارد.



شکل ۴۴.۳: دیاگرام UC داشبورد فریلنسر

۱.۷.۳ ایجاد رزومه

مورد استفاده: ایجاد رزومه

شرح مختصر UC: در این قسمت فریلنسر رزومه خود را ایجاد می‌کند.

پیش شرط: ورود به مدیریت رزومه در داشبورد فریلنسر.

سناریو اصلی:

۱. شروع

۲. فریلنسر دکمه ایجاد رزومه را انتخاب می‌کند و سیستم فرم خام را به فریلنسر نمایش می‌دهد.

۳. فریلنسر فرم را تکمیل می‌کند و با دکمه ارسال، فرم تکمیل شده را به سیستم ارسال می‌کند.

۴. سیستم اطلاعات فرم را بررسی می‌کند و اطلاعات را در بانک اطلاعات ثبت می‌کند.

۵. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در اطلاعات فرم ایجاد رزومه

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در اطلاعات فرم اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به فریلنسر نمایش داده می‌شود و درخواست اصلاح اطلاعات فرم را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: رزومه با موفقیت ایجاد شود

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت آمیز بودن ایجاد رزومه اجرا می‌شود.

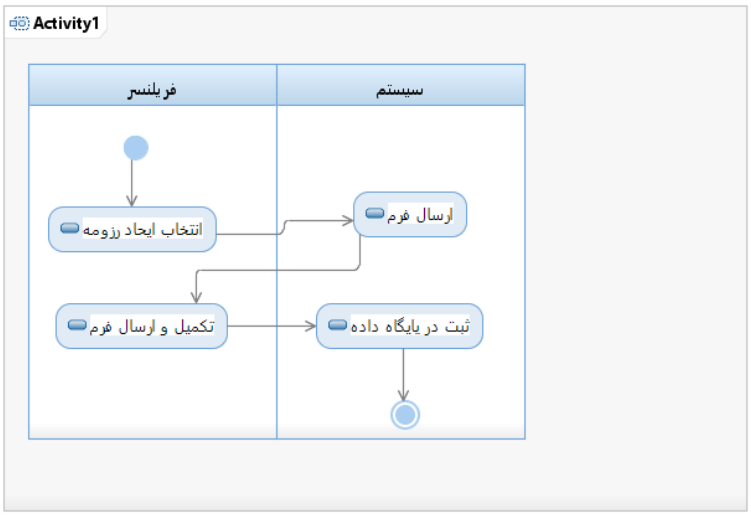
۱. شروع

۲. اطلاعات فرم بررسی می‌شود و یک پیغام به فریلنسر نمایش داده می‌شود که اطلاعات با موفقیت ثبت شده است.

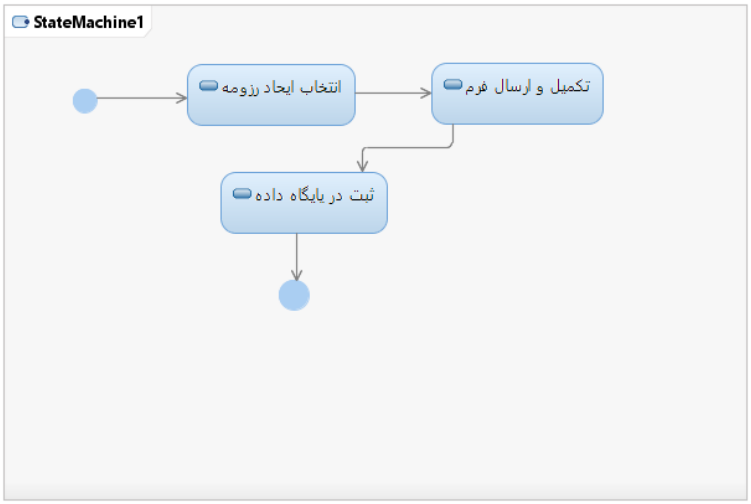
۳. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

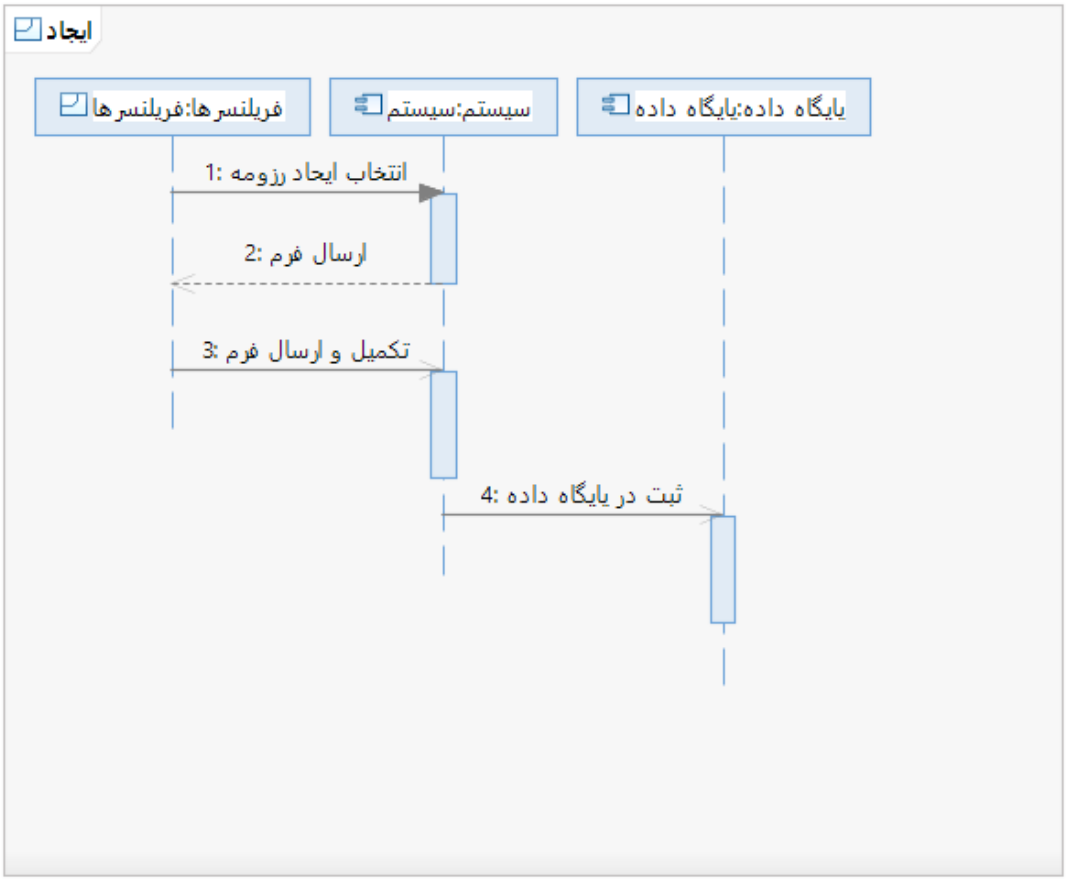
پس شرط: ندارد.



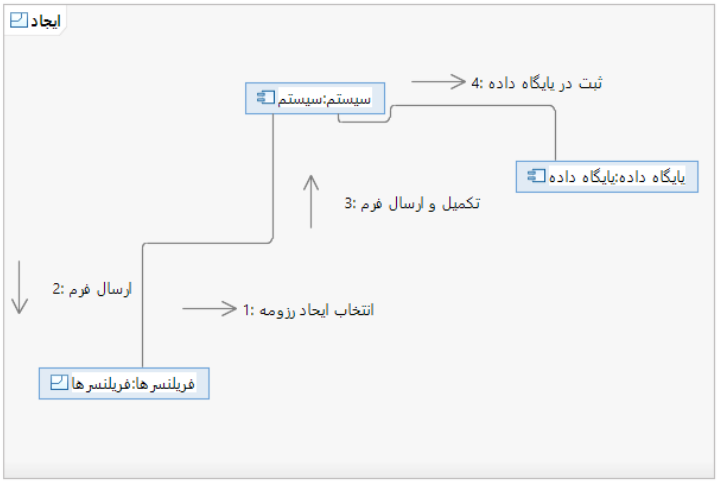
شکل ۴۵.۳: دیاگرام فعالیت ایجاد رزومه



شکل ۴۶.۳: دیاگرام حالت ماشین ایجاد رزومه



شکل ۴۷.۳: دیاگرام توالی ایجاد رزومه



شکل ۴۸.۳: دیاگرام همکار ایجاد رزومه

۲.۷.۳ ویرایش رزومه

مورد استفاده: ویرایش رزومه

شرح مختصر UC: در این قسمت فریلنسر رزومه خود را ویرایش می‌کند.

پیش شرط: ورود به مدیریت رزومه در داشبورد فریلنسر.

سناریو اصلی:

۱. شروع

۲. فریلنسر دکمه ویرایش رزومه را انتخاب می‌کند و سیستم فرم اطلاعات رزومه را به فریلنسر نمایش می‌دهد.

۳. فریلنسر فرم را اصلاح می‌کند و با دکمه ارسال، فرم اصلاح شده را به سیستم ارسال می‌کند.

۴. سیستم اطلاعات فرم را بررسی می‌کند و اطلاعات را در بانک اطلاعات بروزرسانی می‌کند.

۵. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در اطلاعات فرم رزومه

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در اطلاعات فرم اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به فریلنسر نمایش داده می‌شود و درخواست اصلاح اطلاعات فرم را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: رزومه با موفقیت ویرایش شود

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت آمیز بودن ویرایش رزومه اجرا می‌شود.

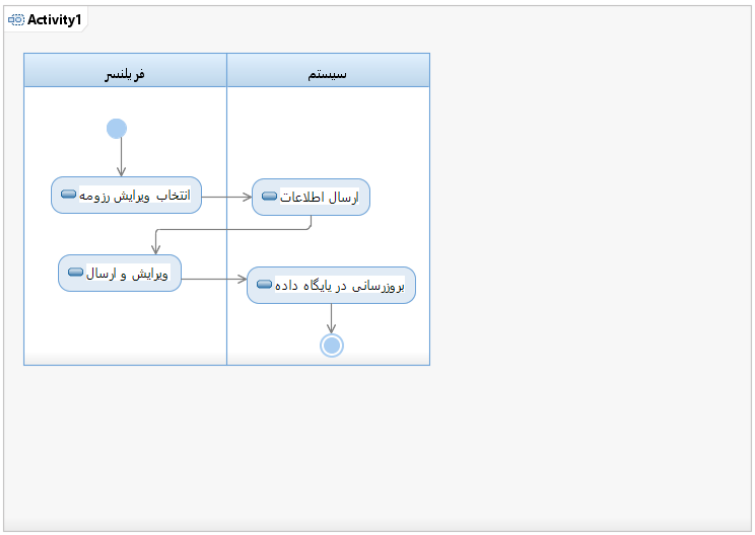
۱. شروع

۲. اطلاعات فرم بررسی می‌شود و یک پیغام به فریلنسر نمایش داده می‌شود که اطلاعات با موفقیت ثبت شده است.

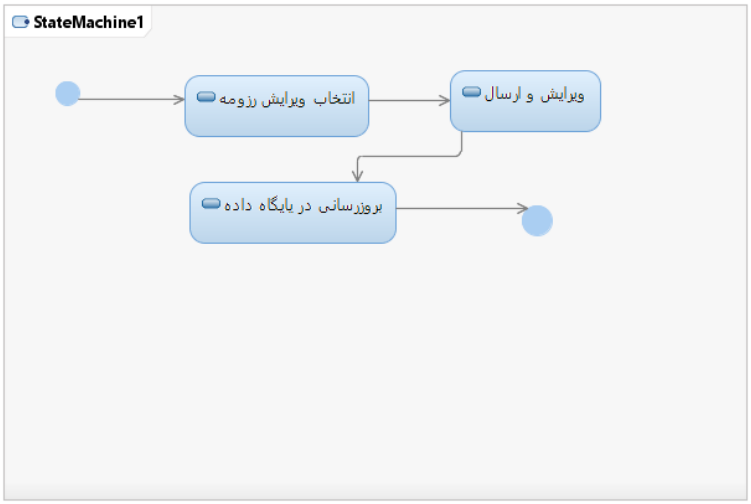
۳. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

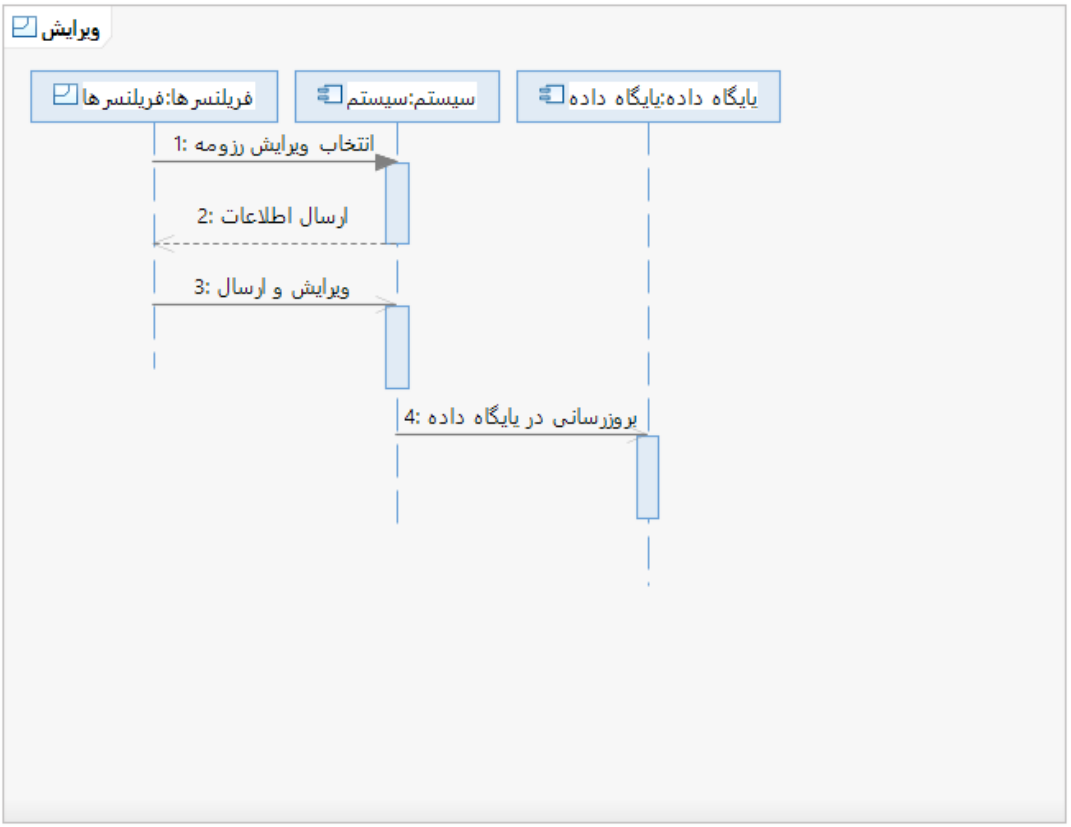
پس شرط: ندارد.



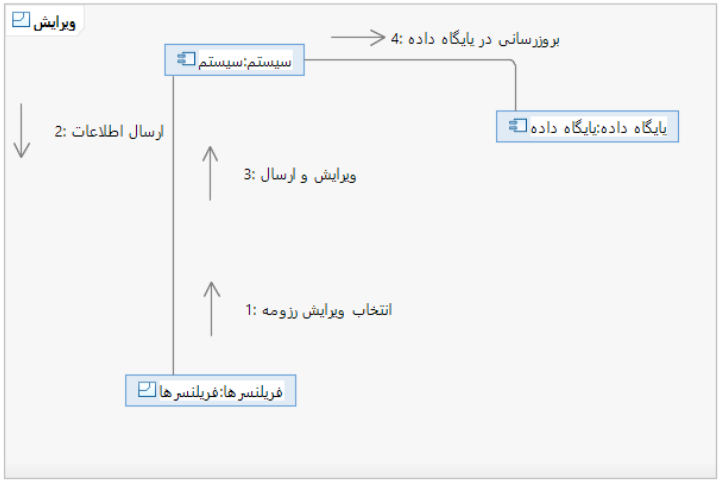
شکل ۴۹.۳: دیاگرام فعالیت ویرایش رزومه



شکل ۵۰.۳: دیاگرام حالت ماشین ویرایش رزومه



شکل ۵۱.۳: دیاگرام توالی ویرایش رزومه



شکل ۵۲.۳: دیاگرام همکار ویرایش رزومه

۳.۷.۳ حذف رزومه

مورد استفاده: حذف رزومه

شرح مختصر UC: در این قسمت فریلنسر رزومه خود حذف می‌کند.

پیش شرط: ورود به مدیریت رزومه در داشبورد فریلنسر.

سناریو اصلی:

۱. شروع

۲. فریلنسر دکمه حذف رزومه را انتخاب می‌کند و سیستم اطلاعات را به فریلنسر نمایش می‌دهد.

۳. فریلنسر تایید حذف رزومه را به سیستم ارسال می‌کند.

۴. سیستم رزومه را بررسی و از بانک اطلاعات حذف می‌کند.

۵. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: رزومه با موفقیت حذف شود

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت آمیز بودن حذف رزومه اجرا می‌شود.

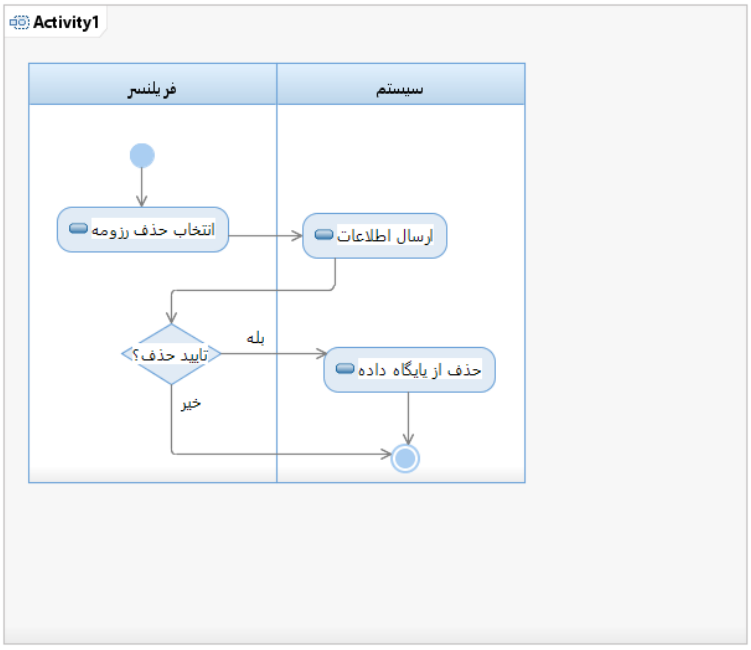
۱. شروع

۲. اطلاعات فرم بررسی می‌شود و یک پیغام به فریلنسر نمایش داده می‌شود که اطلاعات با موفقیت حذف شده است.

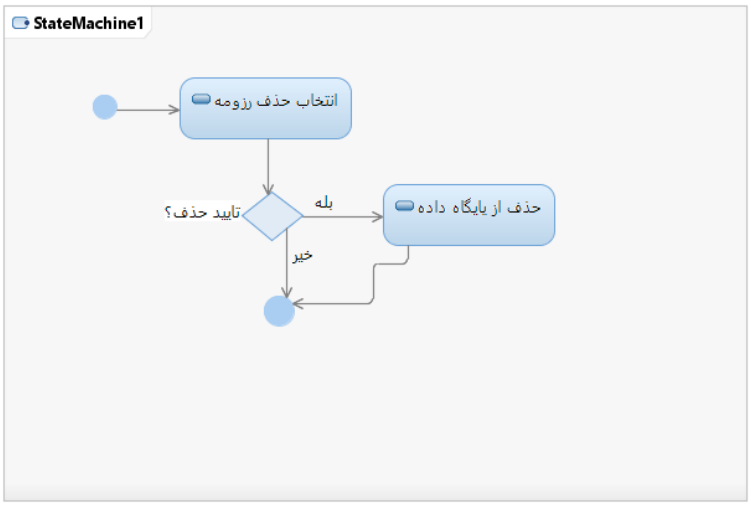
۳. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

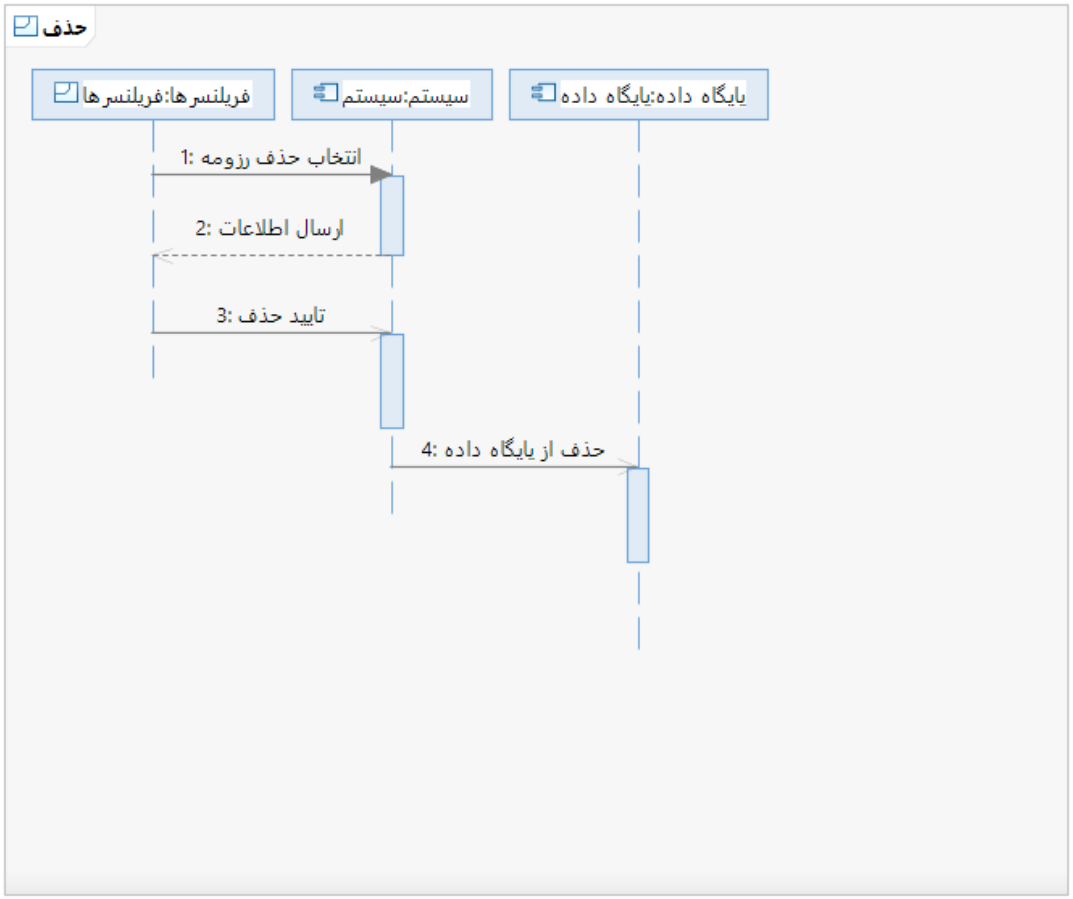
پس شرط: ندارد.



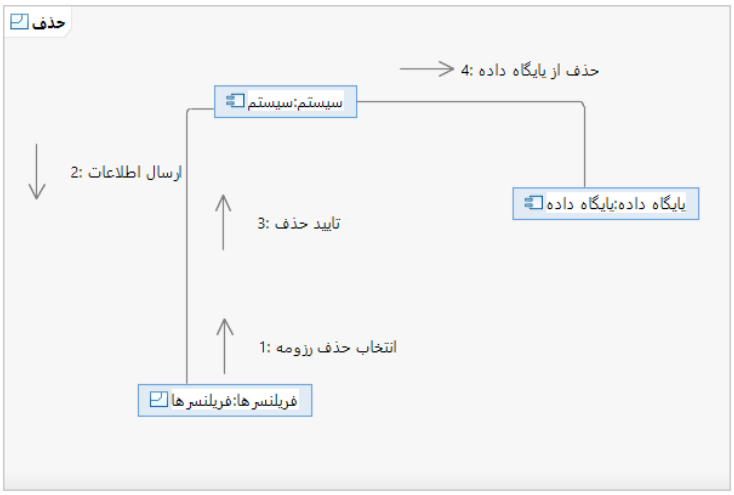
شکل ۵۳.۳: دیاگرام فعالیت حذف رزومه



شکل ۵۴.۳: دیاگرام حالت ماشین حذف رزومه



شکل ۵۵.۳: دیاگرام توالی حذف رزومه



شکل ۵۶.۳: دیاگرام همکار حذف رزومه

۴.۷.۳ درخواست پروژه

مورد استفاده: درخواست پروژه

شرح مختصر UC: در این قسمت فریلنسر پیشنهادات خود برای انجام پروژه را برای کارفرما ارسال می‌کند.

پیش شرط: ورود به مدیریت پروژه در داشبورد فریلنسر.

سناریو اصلی:

۱. شروع

۲. فریلنسر دکمه درخواست پروژه را انتخاب می‌کند و سیستم فرم خام پیشنهاد به کارفرما را نمایش می‌دهد

۳. فریلنسر فرم را تکمیل می‌کند و با دکمه ارسال، اطلاعات را به سیستم ارسال می‌کند.

۴. سیستم اطلاعات را بررسی می‌کند و در بانک اطلاعات ثبت می‌کند.

۵. پایان

پس شرط: کارفرما باید درخواست فریلنسر را تایید کند.

سناریوهای فرعی:

سناریو فرعی ۱: خطا در اطلاعات فرم درخواست پروژه

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت خطا در اطلاعات فرم اجرا می‌شود.

۱. شروع

۲. اطلاعات فرم بررسی می‌شود و خطاها مشخص می‌شوند.

۳. یک پیغام به فریلنسر نمایش داده می‌شود و درخواست اصلاح اطلاعات فرم را دارد.

۴. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

سناریو فرعی ۲: درخواست با موفقیت ثبت شود

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت موفقیت آمیز بودن ثبت درخواست پروژه اجرا می‌شود.

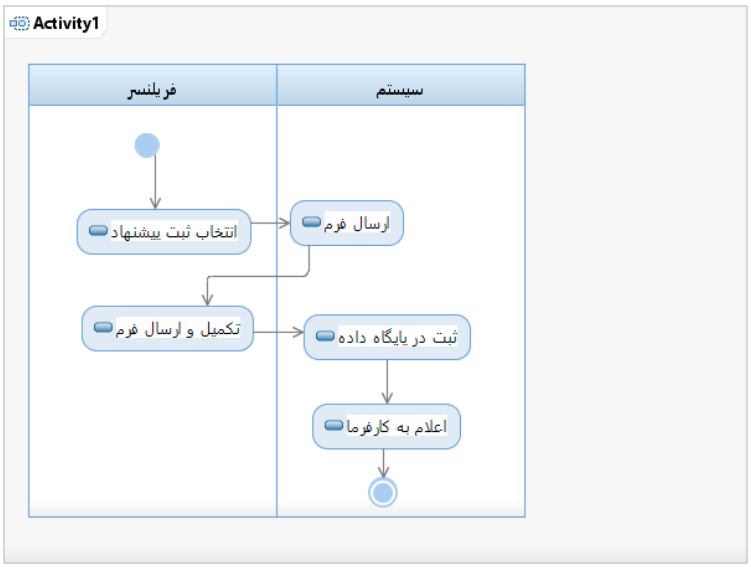
۱. شروع

۲. اطلاعات فرم بررسی می‌شود و یک پیغام به فریلنسر نمایش داده می‌شود که اطلاعات با موفقیت ثبت شده است.

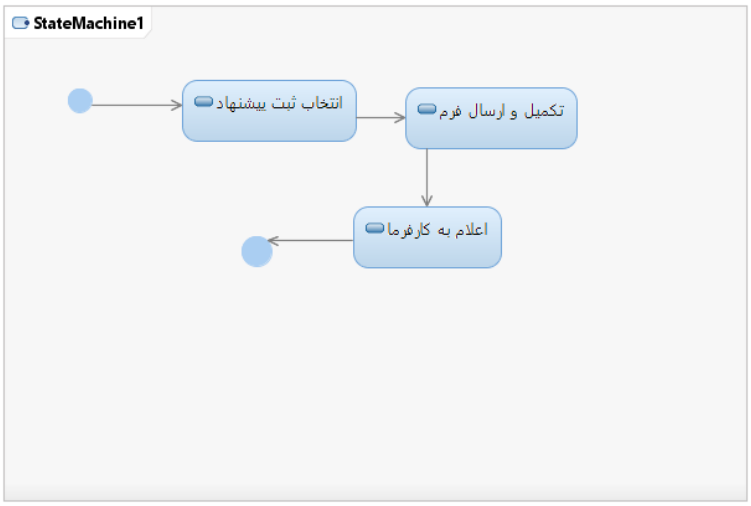
۳. از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

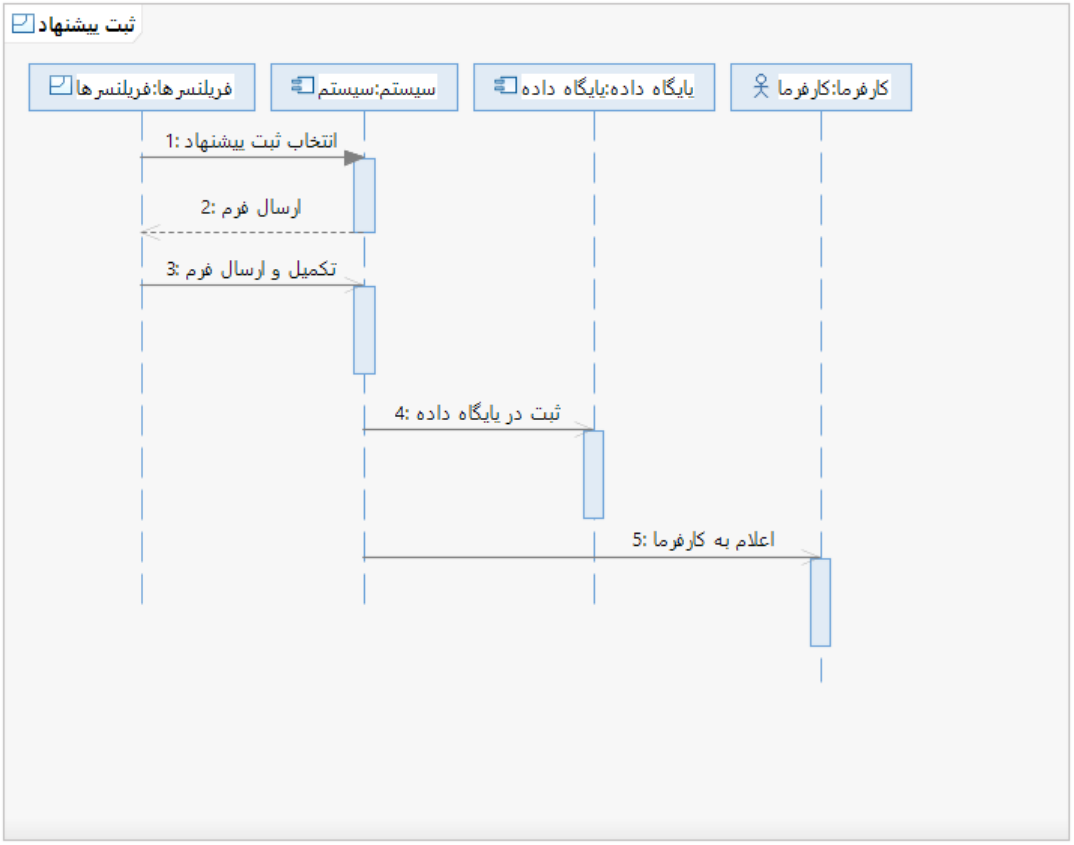
پس شرط: ندارد.



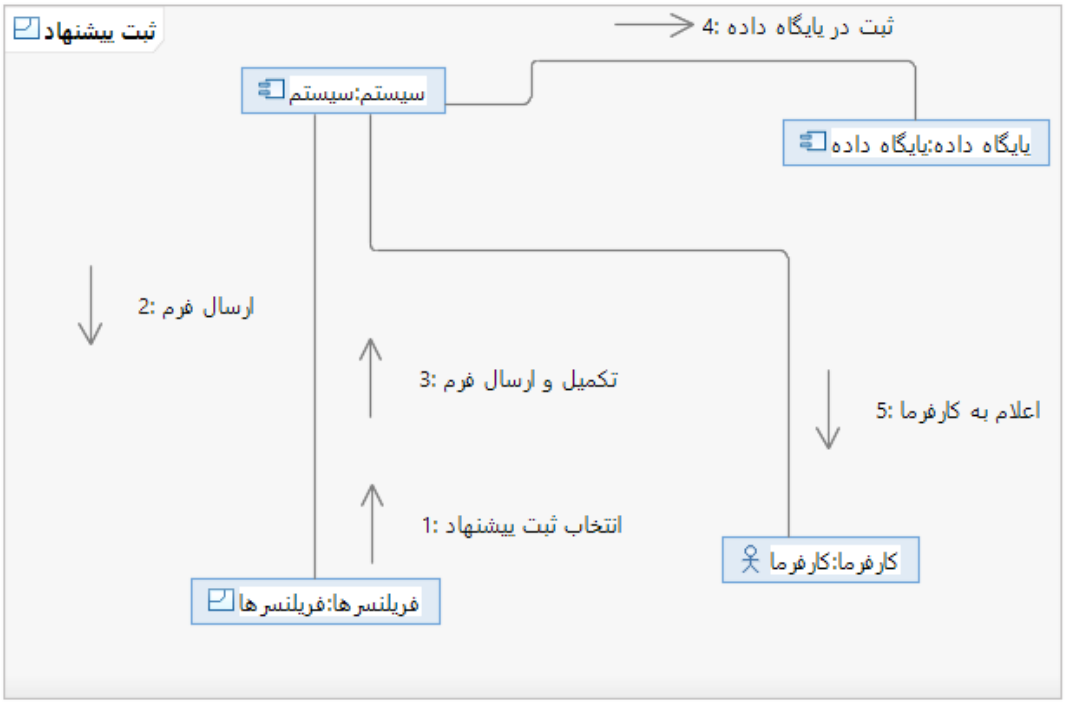
شکل ۵۷.۳: دیاگرام فعالیت درخواست پروژه



شکل ۵۸.۳: دیاگرام حالت ماشین درخواست پروژه



شکل ۵۹.۳: دیاگرام توالی درخواست پروژه



شکل ۶۰.۳: دیاگرام همکار درخواست پروژه

۵.۷.۳ دریافت هزینه پروژه

مورد استفاده: دریافت هزینه انجام پروژه

شرح مختصر UC: در این قسمت فریلنسر هزینه پروژه را دریافت می‌کند.

پیش شرط: ورود به مدیریت مالی در داشبورد فریلنسر.

سناریو اصلی:

۱. شروع

۲. فریلنسر دکمه دریافت هزینه پروژه را انتخاب می‌کند و پروژه انجام شده را در سایت بارگذاری می‌کند.

۳. کارفرما پروژه را تست/رویت و نظر خود را با دکمه ثبت نظر ثبت می‌کند.

۴. پس از تایید/لغو طرفین، پول بلوکه شده آزاد و به حساب فریلنسر/کارفرما واریز می‌شود.

۵. پروژه خاتمه یافتنه و در بانک اطلاعات ثبت می‌شود.

۶. پایان

پس شرط: ندارد.

سناریوهای فرعی:

سناریو فرعی ۱: اصلاحات پروژه

شرح مختصر UC: این سناریو در مرحله ۳ سناریو اصلی کارفرما بخش‌های نیاز به اصلاح را به فریلنسر اعلام می‌کند.

۱. شروع

۲. اصلاحات مورد نظر کارفرما طبق جهارچوب و قوانین سایت به فریلنسر جهت اعمال اعلام می‌شود.

۳. از مرحله ۳ سناریو اصلی ادامه پیدا می‌کند.

۴. پایان

سناریو فرعی ۲: شکایت

شرح مختصر UC: این سناریو در مرحله ۴ سناریو اصلی در صورت شکایت طرفین اجرا می‌شود.

۱. شروع

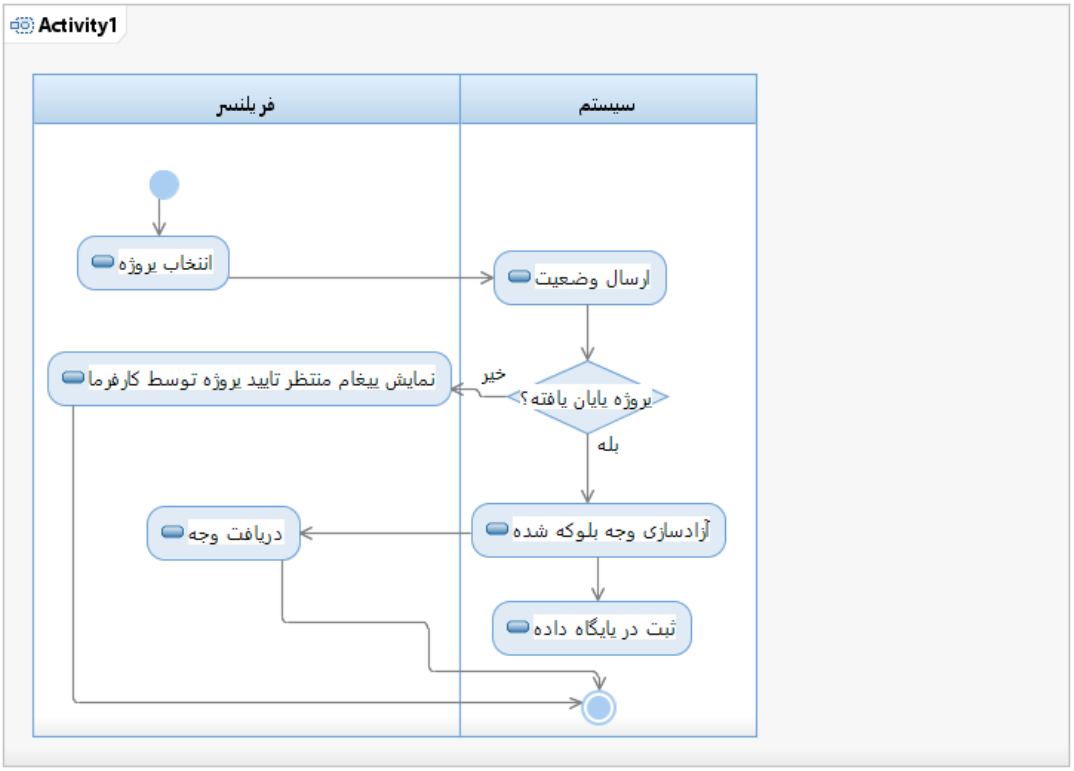
۲. یکی از طرفین دکمه ثبت شکایت را انتخاب می‌کند و توضیحات شکایت خود را ثبت می‌کند.

۳. کارشناسان سایت به پروژه ورود کرده و نظر خود را در رابطه با آن اعلام می‌کنند.

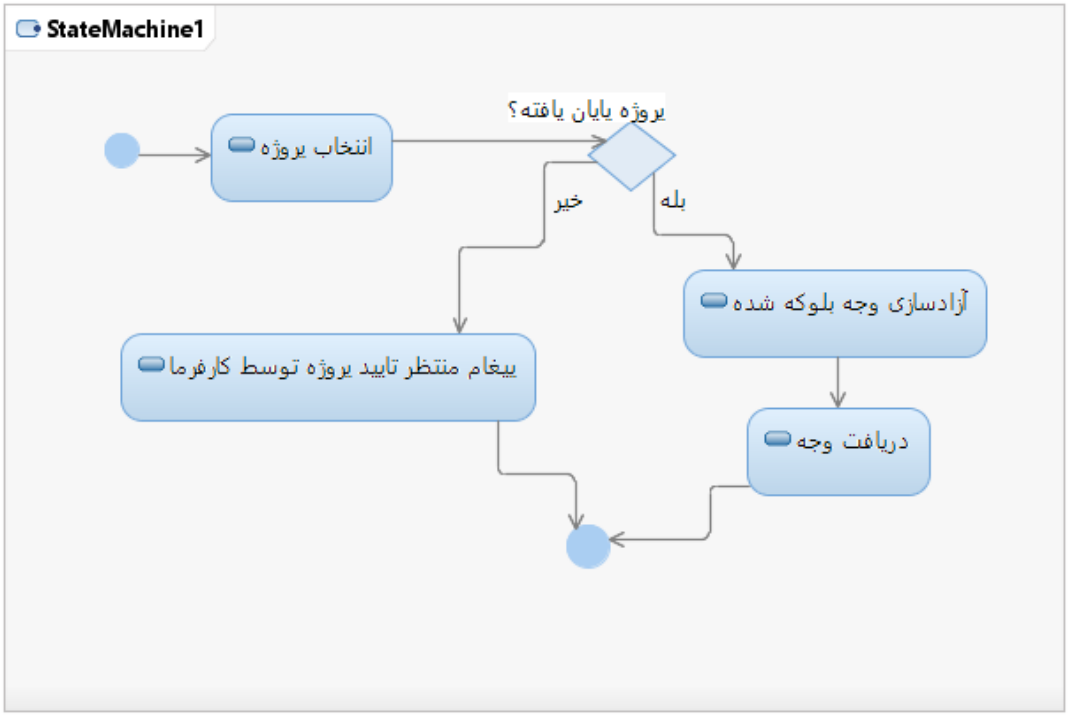
۴. پس از لغو/تایید پروژه، از مرحله ۴ سناریو اصلی ادامه پیدا می‌کند.

۵. پایان

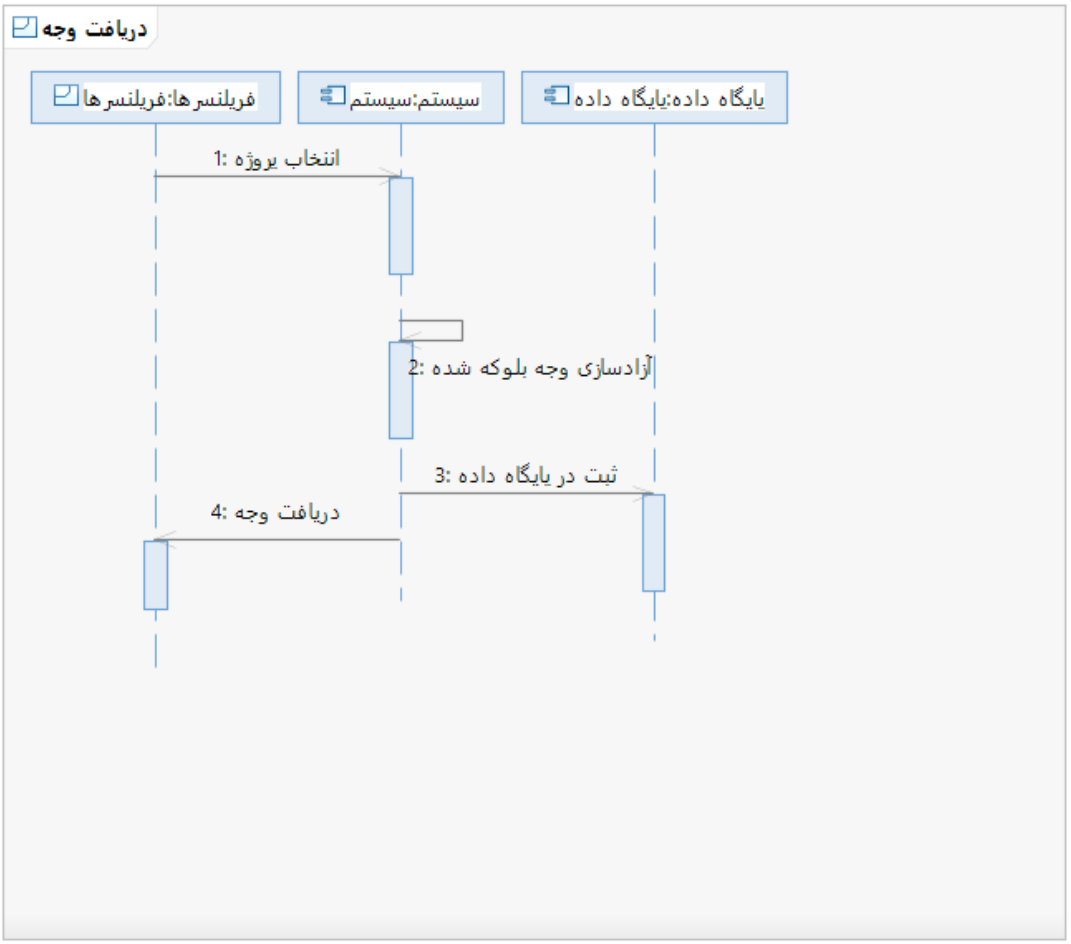
پس شرط: ندارد.



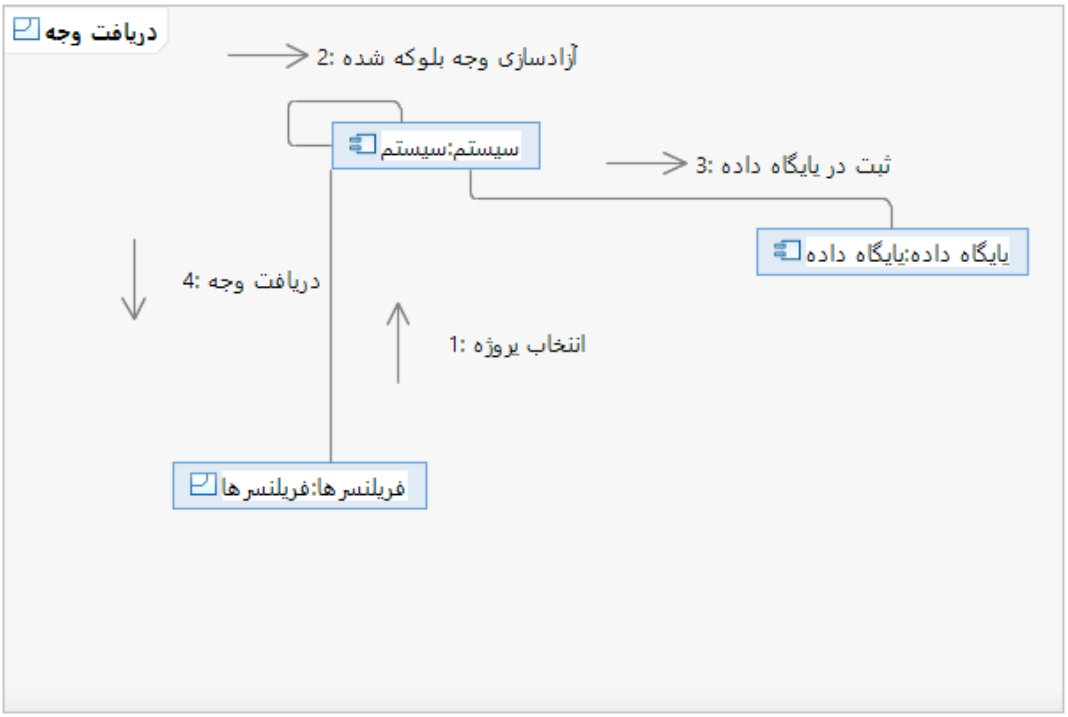
شکل ۶۱.۳: دیاگرام فعالیت دریافت وجه



شکل ۶۲.۳: دیاگرام حالت ماشین دریافت وجه



شکل ۶۳.۳: دیاگرام توالی دریافت وجه



شکل ۶۴.۳: دیاگرام همکار دریافت وجه

۸.۳ داشبورد مدیریت

مورد استفاده: داشبورد مدیریت

شرح مختصر UC: در این قسمت داشبورد مدیریت را در اختیار کاربر قرار می‌دهد.

پیش شرط: ورود کاربر با سطح دسترسی مدیر.

سناریو اصلی:

۱. شروع

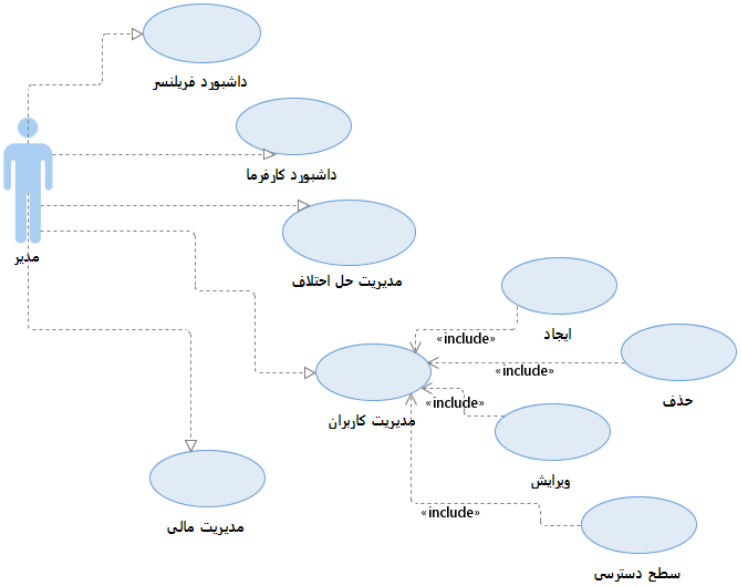
۲. مدیر به بخش‌های مختلف مانند ایجاد و حذف کاربر، تعیین سطوح دسترسی کاربران و .. دسترسی پیدا می‌کند.

۳. پایان

پس شرط: ندارد.

سناریوهای فرعی: ندارد.

پس شرط: ندارد.



شکل ۶۵.۳: دیاگرام UC داشبورد مدیریت

۱.۸.۳ مدیریت مالی

مورد استفاده: مدیریت مالی

شرح مختصر UC: در این قسمت مدیر می‌تواند تمام اطلاعات مالی را مشاهده کند.

پیش شرط: ورود با کاربری مدیر

سناریو اصلی:

۱. شروع

۲. مدیر دکمه مدیریت مالی را انتخاب می‌کند و گردش مالی، تاریخچه، ... را روئیت می‌کند.

۳. پایان

پس شرط: ندارد.

سناریوهای فرعی:

ندارد

۲.۸.۳ مدیریت اختلافات

مورد استفاده: مدیریت اختلافات

شرح مختصر UC: در این قسمت مدیر می‌تواند تمام اختلافات را مشاهده و بررسی کند.

پیش شرط: ورود با کاربری مدیر

سناریو اصلی:

۱. شروع

۲. مدیر دکمه مدیریت اختلافات را انتخاب می‌کند و طرفین درگیر، شکایات، ... را روئیت می‌کند.

۳. پایان

پس شرط: ندارد.

سناریوهای فرعی:

ندارد

۳.۸.۳ مدیریت کاربران

مورد استفاده: مدیریت کاربران

شرح مختصر UC: در این قسمت مدیر می‌تواند تمام کاربران را مشاهده و بررسی کند.

پیش شرط: ورود با کاربری مدیر

سناریو اصلی:

۱. شروع

۲. مدیر دکمه مدیریت کاربران را انتخاب می‌کند و ایجاد، حذف، تعیین سطح دسترسی را روئیت می‌کند.

۳. پایان

پس شرط: ندارد.

سناریوهای فرعی:

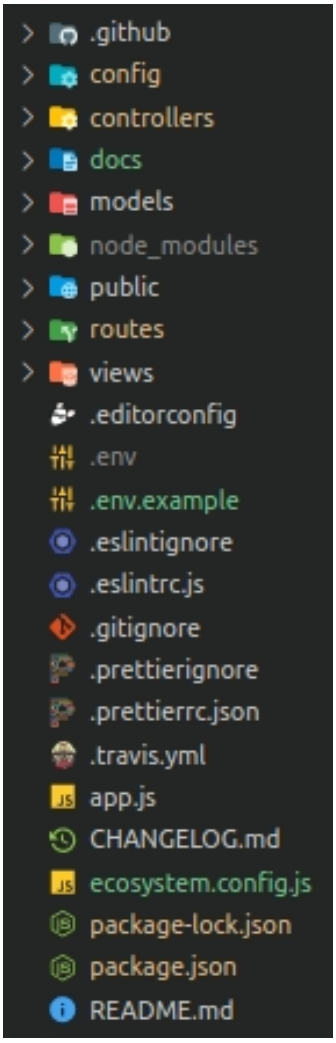
ندارد

فصل۴

طراحی و پیاده‌سازی نرم‌افزار

مقدمه

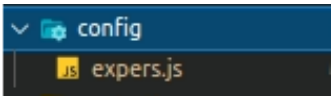
در این فصل به نحوه طراحی و پیاده سازی نرم‌افزار پرداخته شده است.



شکل ۱.۴: ساختار کلی

۱.۴ پوشه تنظیمات (config)

در این پوشه تنظیمات نرم‌افزار قرار دارد.



شکل ۲.۴: ساختار پوشه تنظیمات

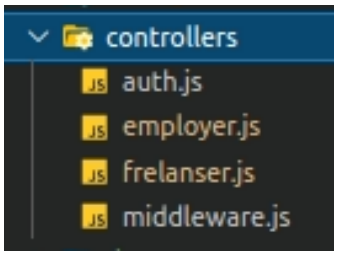
فایل expers در این فایل به تنظیمات فریم‌ورک اکسپرس که شامل ارتباط با پایگاه داده و فراخوانی فایل‌های env. و package پرداخته شده است.

فایل dataBase در این فایل به تنظیمات پایگاه‌داده پرداخته شده است.

فایل variable در این فایل به متغیرهای عمومی نرم‌افزار پرداخته شده است.

۲.۴ پوشه کنترل (controllers)

در این قسمت تمام عملیات‌های پردازش درخواست‌ها شامل ثبت، ویرایش و ... کنترل می‌شوند.



شکل ۳.۴: ساختار پوشه کنترل

۱.۲.۴ فایل middleware

در این فایل به میان‌افزارهای کنترل مانند کنترل سطح دسترسی کاربر، کنترل ورود کاربر و ... پرداخته شده است.

logInChecker: اعتبارسنجی ورود کاربر به سایت.

توضیحات

```
void logInChecker ( object request , object response , object next );
```

بررسی طلاعات و ورود کاربر.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).
next	object	میان‌افزار بعدی را اجرا می‌کند.

خروجی

در صورتی که کاربر به سایت وارد شده باشد، برنامه ادامه می‌یابد، در غیر این صورت به صفحه ورود به سایت هدایت می‌شود.

sessionChecker: اعتبارسنجی نشست‌های برنامه.

توضیحات

```
void sessionChecker ( object request , object response , object next );
```

بررسی اعتبار زمانی.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ HTTP res است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست HTTP req است).
next	object	میان‌افزار بعدی را اجرا می‌کند.

خروجی

در صورتی که مهلت زمانی ورود کاربر به پایان نرسیده باشد برنامه ادامه می‌یابد، در غیر این صورت به صفحه ورود به سایت هدایت می‌شود.

roleChecker: اعتبارسنجی سطح دسترسی کاربر به محتوای برنامه.

توضیحات

```
void logInChecker ( object request , object response , object next );
```

اعتبارسنجی سطح دسترسی کاربر.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ HTTP res است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست HTTP req است).
next	object	میان‌افزار بعدی را اجرا می‌کند.

خروجی

در صورتی که سطح دسترسی کاربر مجاز باشد برنامه ادامه می‌یابد، در غیر این صورت به صفحه قبل از درخواست هدایت می‌شود.

۲.۲.۴ فایل auth

در این فایل به کنترل عملیات‌های احرازهویت پرداخته شده است.

getRegister: اطلاعات صفحه ثبت‌نام را بارگذاری می‌کند.

توضیحات

```
void getRegister ( object request , object response );
```

بارگذاری صفحه ثبت‌نام.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری اطلاعات صفحه ثبت‌نام کاربر.

postRegister: ثبت اطلاعات کاربر در بانک اطلاعات

توضیحات

```
void postRegister ( object request , object response );
```

ثبت اطلاعات کاربر در پایگاه‌داده به عبارت دیگر ایجاد کاربر جدید

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

در صورت موجود نبودن کاربری کاربری جدید ایجاد می‌شود و در غیر این صورت به صفحه ورود ارجاع داده می‌شود.

getLogIn: اطلاعات صفحه ورود را بارگذاری می‌کند.

توضیحات

```
void getLogIn ( object request , object response );
```

هر کاربر با درج اطلاعات صحیح خود با شناسه‌ای یکتا به برنامه وارد می‌شود.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

در صورت موجود بودن کاربری به برنامه وارد می‌شود و در غیراین‌صورت به صفحه ثبت‌نام ارجاع داده می‌شود.

postLogIn: دریافت اطلاعات کاربر از بانک اطلاعات

توضیحات

```
void postLogIn ( object request , object response );
```

دریافت اطلاعات کاربر از پایگاه‌داده و ورود به برنامه

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

در صورت موجود بودن کاربر به صفحه داشبورد ارجاع داده می‌شود و در غیر این صورت به صفحه ثبت‌نام ارجاع داده می‌شود.

getForgot: فراموشی رمز عبور

توضیحات

```
void getForgot ( object request , object response );
```

صفحه فراموشی رمز عبور را بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری صفحه فراموشی رمز عبور.

postForgot: دریافت اطلاعات کاربر از بانک اطلاعات

توضیحات

```
void postForgot ( object request , object response );
```

دریافت اطلاعات کاربر از پایگاه‌داده و ورود به مرحله بازیابی رمز عبور.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

در صورت موجود بودن کاربری به صفحه بازیابی اطلاعات ارجاع داده می‌شود، در غیراین‌صورت به صفحه ثبت‌نام ارجاع داده می‌شود.

getRecover: بازیابی رمز عبور

توضیحات

```
void getRecover ( object request , object response );
```

صفحه بازیابی رمز عبور را بارگذاری و توکن ارسال شده به پست الکترونیک کاربر را اعتبار سنجی می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

درصورت صحت توکن کاربر صفحه بازیابی رمز عبور را بارگذاری می‌کند.

postRecover: ویزایش اطلاعات کاربری بازیابی شده

توضیحات

```
void postRecover ( object request , object response );
```

ویزایش اطلاعات کاربری بازیابی شده در بانک اطلاعات

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

اطلاعات کاربر بازیابی شده و به صفحه داشبورد ارجاع داده می‌شود.

۳.۲.۴ فایل employer

در این فایل به کنترل تمام عملیات‌های داشبورد کارفرما پرداخته شده است.

getRoot: داشبورد کارفرما

توضیحات

```
void getRoot ( object request , object response );
```

صفحه داشبورد کارفرما را بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری داشبورد کارفرما.

getProject: لیست پروژه‌های کارفرما

توضیحات

```
void getProject ( object request , object response );
```

صفحه لیست پروژه‌های کارفرما را بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری صفحه لیست پروژه‌های کارفرما.

getAddProject: تعریف پروژه توسط کارفرما

توضیحات

```
void getAddProject ( object request , object response );
```

صفحه ایجاد پروژه را بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری صفحه ایجاد پروژه توسط کارفرما.

postAddProject: ثبت اطلاعات پروژه در بانک اطلاعات

توضیحات

```
void postAddProject ( object request , object response );
```

ثبت اطلاعات پروژه در پایگاه‌داده

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

اطلاعات پروژه ثبت شده و به صفحه لیست پروژه‌ها ارجاع داده می‌شود.

getEditProject: ویرایش پروژه

توضیحات

```
void getEditProject ( object request , object response );
```

بارگذاری صفحه ویرایش پروژه بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری صفحه ویرایش پروژه توسط کارفرما.

postEditProject: ویرایش اطلاعات پروژه در بانک اطلاعات

توضیحات

```
void postEditProject ( object request , object response );
```

ویرایش اطلاعات پروژه در پایگاه‌داده

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

اطلاعات پروژه ویرایش شده و به صفحه لیست پروژه‌ها ارجاع داده می‌شود.

getDeleteProject: حذف پروژه

توضیحات

```
void getDeleteProject ( object request , object response );
```

حذف پروژه از پایگاه‌داده.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

حذف پروژه از پایگاه‌داده توسط کارفرما.

getDetailProject: نمایش پروژه

توضیحات

```
void getDetailProject ( object request , object response );
```

صفحه نمایش اطلاعات پروژه بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری صفحه نمایش اطلاعات پروژه.

getInvoiceProject: پرداخت هزینه پروژه

توضیحات

```
void getInvoiceProject ( object request , object response );
```

صفحه پرداخت هزینه پروژه بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری صفحه پرداخت هزینه پروژه.

postInvoiceProject: ثبت اطلاعات پرداخت هزینه پروژه در بانک اطلاعات

توضیحات

```
void postInvoiceProject ( object request , object response );
```

ثبت اطلاعات پرداخت هزینه پروژه در پایگاه‌داده

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

ثبت اطلاعات پرداخت هزینه و انجام فرایند تسویه حساب با فریلنسر.

۴.۲.۴ فایل frelanser

در این فایل به کنترل تمام عملیات‌های داشبورد فریلنسر پرداخته شده است.

getRoot: داشبورد فریلنسر

توضیحات

```
void getRoot ( object request , object response );
```

صفحه داشبورد فریلنسر را بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری داشبورد فریلنسر.

getAddRequest: درج پیشنهاد برای پروژه

توضیحات

```
void getAddRequest ( object request , object response );
```

درج پیشنهاد برای پروژه توسط فریلنسر

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری صفحه درج پیشنهاد برای پروژه توسط فریلنسر.

postAddRequest: ثبت اطلاعات پیشنهاد در بانک اطلاعات

توضیحات

```
void postAddRequest ( object request , object response );
```

ثبت اطلاعات پیشنهاد در پایگاه‌داده

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

اطلاعات پیشنهاد فریلنسر ثبت شده و به صفحه لیست پروژه ارجاع داده می‌شود.

getEditRequest: ویرایش پیشنهاد

توضیحات

```
void getEditRequest ( object request , object response );
```

صفحه ویرایش پیشنهاد ثبت شده برای پروژه بارگذاری می‌کند.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری صفحه ویرایش پیشنهاد توسط فریلنسر.

postEditRequest: ویرایش اطلاعات پیشنهاد در بانک اطلاعات

توضیحات

```
void postEditRequest ( object request , object response );
```

ویرایش اطلاعات پیشنهاد ثبت شده برای پروژه در پایگاه‌داده

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

اطلاعات پیشنهاد ویرایش شده و به صفحه لیست پیشنهادها ارجاع داده می‌شود.

getDelRequest: حذف پیشنهاد

توضیحات

```
void getDelRequest ( object request , object response );
```

حذف پیشنهاد ثبت شده برای پروژه از پایگاه‌داده.

پارامترها

پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

حذف پیشنهاد از پایگاه‌داده توسط فریلنسر.

getProfile: نمایش رزومه فریلنسر

توضیحات

```
void getProfile ( object request , object response );
```

صفحه رزومه فریلنسر را بارگذاری می‌کند.

پارامترها

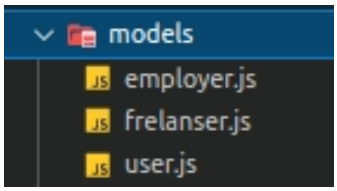
پارامتر	نوع	توضیحات
request	object	نمایانگر درخواست HTTP و دارای خصوصیاتی برای درخواست رشته پرس‌وجو، پارامترها ، بدنه ، هدرهای HTTP و غیره است. در این اسناد و طبق قرارداد ، از این شی همیشه به عنوان req یاد می شود (و پاسخ res HTTP است).
response	object	نمایانگر پاسخ HTTP که برنامه Express با دریافت درخواست HTTP ارسال می کند. در این اسناد و طبق قرارداد ، از شی همیشه به عنوان res یاد می شود (و درخواست req HTTP است).

خروجی

بارگذاری رزومه فریلنسر.

۳.۴ پوشه مدل (models)

در این پوشه به بانک اطلاعات پرداخته می‌شود و تمام اطلاعات از پایگاه داده واکشی می‌شود.



شکل ۴.۴: ساختار پوشه مدل

۱.۳.۴ فایل user

در این فایل به بانک اطلاعات برای ثبت اطلاعات کاربر از جمله مشخصات شناسنامه‌ای، اطلاعات کاربری و رمز و ... پرداخته شده است.

setUser: ثبت اطلاعات کاربر.

توضیحات

```
void setUser ( object newUser , object callback );
```

ثبت طلاعات کاربر در بانک اطلاعات.

پارامترها

پارامتر	نوع	توضیحات
newUser	object	.
callback	object	.

خروجی

در صورتی که ، در غیر این صورت .

getUserByEmail: دریافت اطلاعات کاربر با پست الکترونیک.

توضیحات

```
void getUserByEmail ( string email , object callback );
```

دریافت طلاعات کاربر از بانک اطلاعات توسط پست الکترونیک.

پارامترها

پارامتر	نوع	توضیحات
email	string	.
callback	object	.

خروجی

در صورتی که ، در غیر این صورت .

getUserById: دریافت اطلاعات کاربر با .ID

توضیحات

```
void getUserById ( int id , object callback );
```

دریافت طلاعات کاربر از بانک اطلاعات توسط .ID

پارامترها

پارامتر	نوع	توضیحات
id	int	.
callback	object	.

خروجی

در صورتی که ، در غیر این صورت .

۲.۳.۴ فایل employer

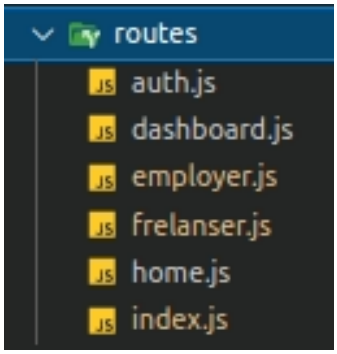
در این فایل به بانک اطلاعات برای اطلاعات داشبورد کارفرما مانند ثبت و بروزرسانی پروژه، دریافت پیشنهادهای انجام پروژه و ... پرداخته شده است.

۳.۳.۴ فایل frelanser

در این فایل به بانک اطلاعات برای اطلاعات داشبورد فریلنسر مانند ثبت رزومه، ثبت پیشنهاد برای انجام پروژه و ... پرداخته شده است.

۴.۴ پوشه مسیر (routes)

در این پوشه به تمام مسیرهای نرم‌افزار پرداخته شده است.



شکل ۵.۴: ساختار پوشه مسیر

۱.۴.۴ فایل index

در این فایل مسیرهای برنامه تجمیع شده و تنها این فایل برای دسترسی به مسیرها فراخوانی می‌شود.

infoApp: شئ حاوی اطلاعات عمومی مانند اطلاعات کاربر، اطلاعات شاخه مسیرها و ... است.

۲.۴.۴ فایل home

در این فایل به مسیرهای صفحه اصلی، لیست کاربران و لیست پروژه‌ها پرداخته شده است.

CHome: شئ حاوی اطلاعات فایل home در پوشه کنترل‌گر است.

۳.۴.۴ فایل auth

در این فایل به مسیرهای احرازهویت شامل ورود، ثبت‌نام و خروج کاربر پرداخته شده است.

CAuth: شئ حاوی اطلاعات فایل auth در پوشه کنترل‌گر است.

۴.۴.۴ فایل dashboard

در این فایل به مسیرهای داشبورد کاربر پرداخته شده است.

CDashboard: شئ حاوی اطلاعات فایل dashboard در پوشه کنترل‌گر است.

۵.۴.۴ فایل employer

در این فایل به مسیرهای داشبورد کارفرما پرداخته شده است.

CEmployer: شئ حاوی اطلاعات فایل employer در پوشه کنترل‌گر است.

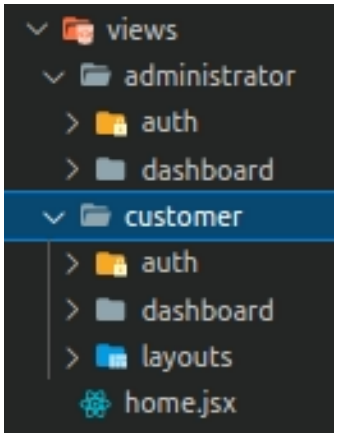
۶.۴.۴ فایل frelanser

در این فایل به مسیرهای داشبورد فریلنسر پرداخته شده است.

CFrelanser: شئ حاوی اطلاعات فایل frelanser در پوشه کنترل‌گر است.

۵.۴ پوشه نما (views)

در این پوشه رابط تعامل کاربر با سیستم انجام می‌شود.



شکل ۶.۴: ساختار پوشه نما

پوشه administrator در این پوشه به قالب مدیریت پرداخته شده است.

پوشه customer در این پوشه به قالب کاربر که شامل کارفرما و فریلنسر پرداخته شده است.

پوشه auth در این پوشه به قالب احرازهویت کاربر پرداخته شده است.

پوشه dashboard در این پوشه به قالب داشبوردهای کارفرما و فریلنسر پرداخته شده است.

پوشه employer در این پوشه به اجرای قالب داشبورد کارفرما پرداخته شده است.

پوشه frelanser در این پوشه به اجرای قالب داشبورد فریلنسر پرداخته شده است.

پوشه component در این پوشه به قالب‌های پویا مانند باکس‌ها، نمودارها، اعلانات و ... پرداخته شده است.

پوشه layouts در این پوشه به قالب تکرارشونده مانند منوها، هدر، فوتر و ... پرداخته شده است.

۶.۴ پوشه عمومی (public)

در این پوشه فایل‌های ثابت در کل برنامه مانند تصاویر، ،js css و ... قرار می‌گیرد.

شکل ۷.۴: ساختار پوشه عمومی

پوشه css در این پوشه خروجی‌های scss قرار می‌گیرد.

پوشه js در این پوشه فایل‌های جاوا اسکریپت قرار می‌گیرد.

پوشه scss در این پوشه فایل‌های scss قرار می‌گیرد.

پوشه plugin در این پوشه پلاگین‌های بکار رفته در برنامه قرار می‌گیرد.

پوشه img در این پوشه تصاویر قرار می‌گیرد.

۷.۴ فایل اصلی/اجرا (app.js)

اجرای نرم‌افزار از این فایل شروع بوده و تمام اطلاعات لازم جهت اجرا فراخوانی می‌شوتد.

۸.۴ پوشه (husky.)

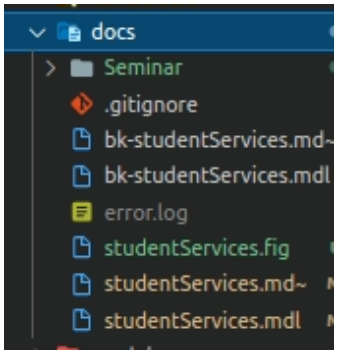
Hook های لازم جهت مدیریت Git در این پوشه قرار می‌گیرد.

۹.۴ پوشه (vscode.)

تنظیمات لازم جهت هماهنگی و مدیریت ادیتور Visual Studio Code در این پوشه قرار می‌گیرد.

۱۰.۴ پوشه مستندات (docs)

در این پوشه تمام مستندات برنامه قرار دارد

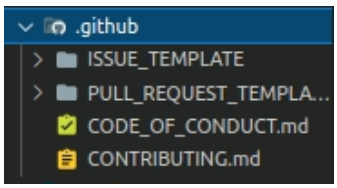


شکل ۸.۴: ساختار پوشه مستندات

پوشه Seminar در این پوشه اطلاعات سمینار و سمینار تتبع با فرمت و قالب LaTeXقرار دارد.

۱۱.۴ پوشه گیت‌هاب (.github)

در این پوشه تمام اطلاعات گیت‌هاب قرار دارد



شکل ۹.۴: ساختار قرارگیری گیت‌هاب

پوشه ISSUE TEMPLATE در این پوشه قالب ایجاد مسئله شامل خطا، باگ و ... در گیت‌هاب قرار دارد.

پوشه PULL REQUEST TEMPLATE در این پوشه قالب ایجاد درخواست ادغام در گیت‌هاب قرار دارد.

فایل CODE OF CONDUCT در این فایل نحوه مشارکت در پروژه تعریف شده است.

فایل CONTRIBUTING در این فایل مشارکت‌کنندگان در پروژه تعریف شده‌اند.

۱۲.۴ فایل Environment (.env)

متغیرهای محیطی بخشی اساسی برای توسعه و کار با Node.js یا هر زبان سمت سرور دیگری می باشد. آنها همیشه حاوی داده های بسیار حساس هستند اما با این تفاوت که نمی خواهند داده هایشان را با دنیای بیرون به اشتراک بگذارند.

۱۳.۴ فایل تاریخچه تغییرات (CHANGELOG)

لاگ تغییرات فایلی است که لیست تغییرات قابل توجه برای هر نسخه یک پروژه که بر اساس تاریخ مرتب شدند را شامل می‌شود

۱۴.۴ فایل Travis CI (.travis)

CI Travis یک سرویس ادغام مداوم توزیع شده و میزبانی شده برای ایجاد و تست پروژه های نرم افزاری میزبانی شده در GitHub می باشد.

۱۵.۴ فایل Editor Config (.editorconfig)

EditorConfig پلاگینی است که به دولوپرها کمک می‌کند تا بتوانند استایل‌های کدنویسی مد نظر خود را در ادیتورها و محیط‌های توسعهٔ یکپارچه (IDE) مختلف حفظ کنند تا از این طریق پس از سوئیچ کردن بین ادیتورهای مختلف، به خاطر اختلاف فضای محیط کدنویسی، دچار سردرگمی نشوند.

۱۶.۴ فایل ESLint (.eslintrc)

ESLint یک Linter برای زبان برنامه‌نویسی Javascript هست که با استفاده از Node.js به وجود اومده است.

همونطور که میدونین Javascript مانند زبانهای دیگه همچون Java و ... نیست و کامپایلری ندارد و کدها مستقیما در مرورگر اجرا میشن. در زبانهای دیگه که کامپایلر وجود دارد، در زمان compile کردن کد، اگر مشکلی وجود داشته باشد در اکثر موارد بیان میشه و compile با موفقیت به پایان نمیرسه ولی در Javascript به دلیل عدم وجود ،compiler مشکلات خودشون رو در زمان اجرا شدن کد در مرورگر نمایش میدن.

ESLint این ارورها رو برای شما پیدا میکنه و جلوی چنین اتفاقاتی رو میگیره. شما بیشتر به دنبال چه نوع ارورهایی هستید که در کدهاتون رخ نده؟

- جلوگیری از حلقه‌های بی‌نهایت یا loop infinite با قرار دادن شرط نامناسب
- اطمینان از اینکه همه متدهای ،getter چیزی رو return میکنند.
- جلوگیری از قرار دادن console.log در کدها
- چک کردن هایcase تکراری در switch
- چک کردن کدهای غیر قابل دسترس. مثلا بعد از return هر کدی رو قرار بدیم، unreachable یا غیر قابل دسترسی میشه.

ESLint بسیار منعطف و با قابلیت تنظیم بالا هست. شما میتونین مشخص کنید که چه rule هایی باید برای کدهای شما چک بشه. همچنین میتونین مشخص کنید که چه نوع استایل استانداردی رو میخواید برای کدهاتون قرار بدین. خیلی

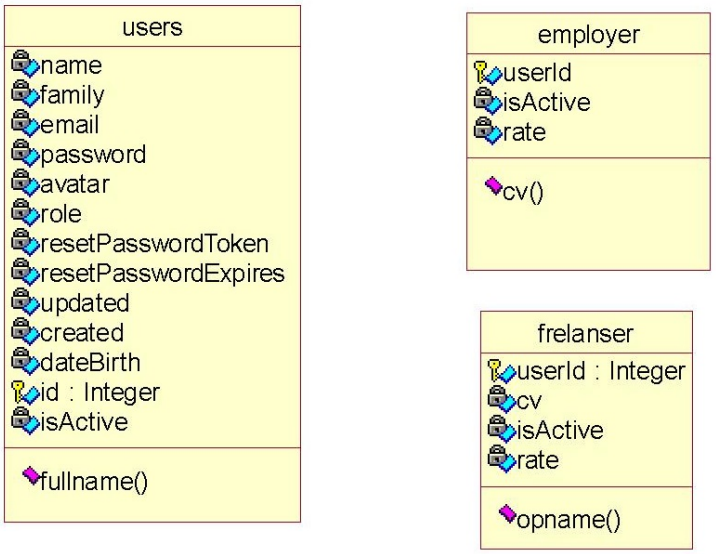
از هاrule بصورت پیش فرض غیر فعال یا فعال هستند و شما میتونین با استفاده از فایل eslintrc. برای کل پروژه‌ها یا یک پروژه خاص، تنظیمات مورد نظرتون رو قرار بدین.

۱۷.۴ فایل Prettier (.prettierrc)

Prettier یک ابزار برای شکل‌دهی به کدها است. در واقع این ابزار کاری می‌کند که کدهای شما در یک قالب منحصر به فرد قرار بگیرند و مرتب شوند.

۱۸.۴ ساختار پایگاه‌داده

در این قسمت به ساختار پایگاه داده پرداخته شده است.



شکل ۱۰.۴: ساختار پایگاه داده

۱.۱۸.۴ پایگاه‌داده کاربر

در این قسمت تمام اطلاعات مربوط به کاربران ثبت می‌شود.

شکل ۱۱.۴: ساختار پایگاه داده کاربر

name: حاوی اطلاعات تام و نام‌خانوادگی.

username: حاوی اطلاعات نام کاربری و تاریخچه تغییرات آن.

email: حاوی اطلاعات پست الکترونیک و تاریخچه تغییرات آن.

password: حاوی اطلاعات رمز عبور و تاریخچه تغییرات آن.

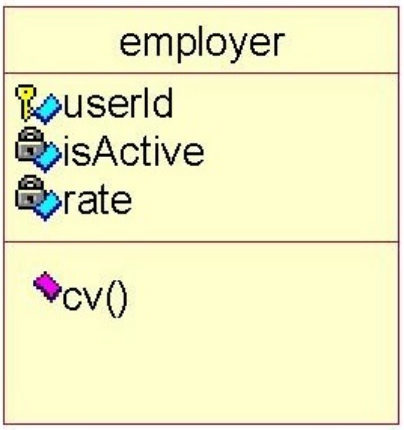
isActive: دسترسی کاربر به داشبوردها را مشخص می‌کند.

avatar: حاوی اطلاعات تصویر پروفایل کاربر.

role: سطح دسترسی کاربر را مشخص می‌کند.

۲.۱۸.۴ پایگاه‌داده کارفرما

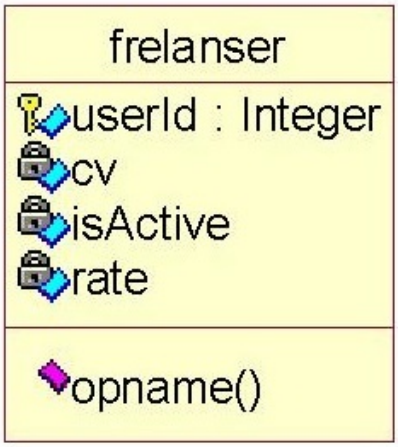
در این قسمت تمام اطلاعات مربوط به ببخش کارفرما ثبت می‌شود.



شکل ۱۲.۴: ساختار پایگاه داده کارفرما

۳.۱۸.۴ پایگاه‌داده فریلنسر

در این قسمت تمام اطلاعات مربوط به بخش فریلنسر ثبت می‌شود.



شکل ۱۳.۴: ساختار پایگاه داده فریلنسر

۴.۱۸.۴ پایگاه‌داده مدیریت

در این قسمت تمام اطلاعات مربوط به بخش مدیریت سایت ثبت می‌شود.

شکل ۱۴.۴: ساختار پایگاه داده مدیریت

فصل۵

جمع‌بندی و پیشنهادات

مقدمه

همانطور که توضیح دادیم، در دنیای امروزه دیگر نیاز نیست برای کسب درآمد به خارج از منزل بروید و دنبال آشنایان خود برای شاغل شدن بگردید. شما می‌توانید بر روی یک مهارت پول‌ساز و مورد علاقه خود تمرکز کنید و پس از کسب مهارت در سایت‌های فریلنسری خارجی و داخلی کسب درآمد کنید. بعضا سایت‌های خارجی فریلنسری گزینه بسیار مناسب برای کسب درآمد دلاری است که شما می‌توانید با بررسی سایت‌های پیشنهاد شده حداقل یک یا دو مورد را انتخاب کرده و با برنامه‌ریزی مناسب و صبر کافی درآمد دلاری کسب کنید.

واژه‌نامه

فارسی	به	انگلیسی

پایگاه‌های داده چند مدله . . . . . Multimodel Database

پایگاه‌های داده گرافی . . . . . Graph Database

پایگاه‌های داده سندی . . . . . Document Database

پایگاه‌های داده ستونی . . . . . Wide-Column Database

پایگاه‌های داده کلید-مقدار . . . . . Key-Value Database

سکو / پلتفرم . . . . . Platform

مجموعه . . . . . D

انگلیسی **به** **فارسی**

مجموعه . . . . . D
مجموعه . . . . . D

Abstract

Computer systems have opened their place among the people of the society during the time they have entered our society. The society has also felt the need for these systems in order to be able to do things faster with these systems. Computer systems also require more advanced software. Therefore، it is up to us to meet these needs by building the necessary software. The student service system can also be one of these softwares to be able to meet some of these needs. On the other hand، to produce these softwares، it is necessary to know the programming language، and NodeJS is one of these languages، which by learning the necessary skills in this The language can easily produce the software needed. The student service system is for defining the project (employer) and carrying out the project (freelancer)، which has features such as project insertion، student admission، search among students and search among projects، and editing and correcting information، and so on. In the past، it was customary to work outside the home to earn a living، and working at home was not privately defined. But recently، in the last few years، with the dramatic increase of the Internet space and in the nature of Internet businesses، family businesses such as freelancing have been introduced. Freelance jobs do not need to be introduced in person، and those who have enough skills in their field will undoubtedly be able to succeed in this field and earn a good income.

Keywords:

Freelancer، Employer، Service Site

Payame Noor University

Faculty of Engineering

Seminar Report

Design and implementation of student services site

Mohammad Amanalikhani

Supervisor:

Dr. Seyed Ali Razavi Ebrahimi

June ۲۰۲۱