موضوع:

سیستم تحت وب پزشکی با امکانات رزرواسیون


بسم الله الرحمن الرحيم

تقدیم به:

تمام رهپویان علم و معرفت

چکیده

سایت پزشکی با امکانات رزرواسیون

هدف این پروژه توسعه نرم افزاری برای یکپارچه سازی سیستم نوبت دهی پزشکی می باشد تا تمامی پزشکان، کلینیک ها و بیماران بتوانند در این سیستم ثبت نام کنند، قرار ملاقات تنظیم کنند و مراحل کار را دنبال کنند. همچنین طراحی واسط کاربر کاربردی برای تنظیم قرار ملاقات و تعقیب ان توسط پزشک از اولویت های این پروژه بشمار می اید. سعی شده است در این پروژه از ابزار ها و تکنولوژی های توسعه نرم افزار بروز استفاده شود و مسائل امنیتی از جنبه های متفاوت مورد توجه قرار گیرد

واژهای کلیدی: رزرواسیون پزشکی، تنظیم و تعقیب قرار ملاقات، سیستم پزشکی، احراز هویت و مجوز، وب سرویس

فهرست مطالب

عنوان
صفحه

فهرست شکل ها

عنوان

صفحه

فهرست جدول ها

عنوان

صفحه

علایم و اختصارات

ماشین مجازی جاوا (java virtual mechine)	JVM
نسخه سازمانی جاوا (java enterprise edition)	JavaEE
نگاشت شئ گرایی به رابطه ای (Object Relational Mapping)	ORM
زبان پرس جو هایبرنت (Hibernate Query Language)	HQL
Model, view, Controller	MVC
Representational State Transfer	REST
Simple Object Access Protocol	SOAP
JSON Web Token	JWT
Single Page Application	SPA

فصل اول: مقدمه

۱-۱- مقدمه

نرم افزار های سنتی بصورت مجزا و منزوی کار میکردند بدین صورت که هر شرکت یا صاحب کسب و کاری نرم افزار مخصوص بخود را داشت و بطور جداگانه از دیگر شرکت ها عمل میکرد ولی امروزه شاهد ان هستیم که سیستم ها به سمت یکپارچه شدن پیش می روند از شرکت های تاکسیرانی گرفته تا شرکت های خدماتی همگی بسرعت به سمت یکپارچگی حرکت میکنند.
سیستم های پزشکی نیز از این قضیه مجزا نیستند و درحال حاضر شاهد توسعه سیستم هایی هستیم که پزشکان و بیماران بتوانند در ان ثبت نام کنند و از امکانات ان استفاده کنند

۱-۲- شرح مسئله

در شکل معمول و سنتی بیماران بصورت تلفنی یا حضوری به پزشکان مراجعه مکنند و از انها نوبت میگیرند. پزشکان نوبت های بیماران را در دفتر مخصوص یا در نرم افزاری نوبت دهی ثبت می کنند. پس اگر بیمار جدید باشد باید مشخصات بیمار را ثبت کنند سپس برایش نوبت بگیرند. اگر بیمار بخواهد نوبت خود را لغو کند و یا جا بجا کند باید با مطب پزشک تماس بگیرد و اطلاع دهد تا پزشک یا منشی او نوبت را لغو کنند.
در این پروژه روند کاری ذکر شده به همراه امکاناتی جهت ساده کردن ان پیاده سازی شده است
این سیستم دارای سه نوع کاربر (نقش) پزشک ، بیمار ، کلینیک می باشد
هر پزشکی که در این سیستم ثبت نام کند پنل کاربری به او داده میشود که به او امکان رزرو یا تغییر نوبت را می دهد همچنین برای پزشک امکانات دیگری مانند تهیه گزارش وجود دارد که میتواند از ان استفاده کند
کلینک ها نیز میتوانند در این سیستم ثبت نام کنند. به هر کلینیک یک پروفایل اختصاص داده میشود تا بتواند از امکانات سیستم استفاده کند
بیماران نیز می توانند در این سیستم ثبت نام کنند و نوبت هایی که برای انها توسط پزشکشان رزرو شده است را مشاهده کنند

۱-۳- اهمیت یکپارچه سازی

پیشرفت جوامع بشری و تحویل کسب و کارهای کوچک به سازمانهای بزرگ با وظایف پیچیده تر بامنابع انسانی زیاد هماهنگی میان اجزا و فرایندهایشان را به چالشی روزافزون مبدل ساخته است امروزه هماهنگی اجزای یک سازمان یگانه عامل جهش و همچنین بقا دررقابت سخت جهانی است باید بدنبال راه حلی بود که باتوجه به شرایط کنونی سازمان علاوه برکمترین اتلاف منابع بیشترین انعطاف ممکن جهت تطبیق با شرایط ناپایدار بازار رقابت را داشته باشد این رویکرد می بایست خطرمقاومت سازمان دربابر تغییروفروپاشی بنیادین آن را درنظر گرفته و همزمان سرعت عملی قابل قبول ازخود به نمایش بگذارد دراین تحقیق به دنبال ارایه راه حلی هستیم تا بتوانیم پس ازمطالعه و درک کل سازمان و اهداف و ماموریت های آن فرایندها را ازابتدا ازکل به جز درنظر گرفته و پس ازحصول اطمینان ازدرک کلیه موانع داخلی و خارجی طراحی مجدد آنها باید این سه اصل ساده سازی استانداردسازی و یکپارچه سازی رامدنظر بگیریم و این بارازجزء به کل برسد ماژولار باشد درهرسه اصل فناوری اطلاعات و سیستم های اطلاعاتی نقش توانمندسازی و کلیدی را ایفا می کنند [1]

۱-۴- اهداف پروژه

از اهداف این پروژه میتوان به موارد زیر اشاره داشت

یکپارچه سازی سیستم نوبت دهی پزشکی

ایجاد محیط ساده و کاربردی پزشک ، بیمار و کلینیک

سرعت بخشیدن به روند تنظیم قرار ملاقات و تعقیب ان

۱-۵- ساختار پروژه

این پایان نامه شامل ۴ فصل می باشد

فصل اول مقدمه ای در باره اهمیت یکپارچه سازی سیستم ها و شرح مسئله موضوع پروژه می باشد

فصل دوم توضیحاتی راجع به ابزار های استفاده شده ، زبان برنامه نویسی ، معماری سیستم و تکنیک های توسعه سیستم داده می شود

در فصل سوم سیستم موضوع پروژه را تحلیل میکنیم و سپس به طراحی سیستم می پردازیم

در فصل چهارم کد های استفاده شده را شرح می دهیم

فصل دوم : اشنایی با ابزار استفاده شده

۲-۱- جاوا ، اسپرینگ ، هایبرنت

جاوا یک زبان برنامه‌نویسیِ شیءگرا است که نخستین‌بار توسط جیمز گاسلینگ در شرکت سان‌مایکروسیستمز ایجاد گردید و در سال ۱۹۹۱ به‌عنوان بخشی از سکوی جاوا منتشر شد.

زبان جاوا، شبیه به سی‌پلاس‌پلاس است، اما مدل شیءگرایی آسان‌تری دارد و از قابلیت‌های سطح پایین کمتری پشتیبانی می‌کند. ایده‌ٔ شیء گرایی جاوا از زبان اسمال‌تاک گرفته شده‌است. یکی از قابلیت‌های بنیادین جاوا این است که مدیریت حافظه را به‌طور خودکار انجام می‌دهد. ضریب اطمینان عملکرد برنامه‌های نوشته‌شده به این زبان نسبت به زبان‌های نسل اول C بالاتر است. برنامه‌های جاوا به صورت بایت کد می‌شوند و توسط ماشین مجازی جاوا به کدهای ماشین تبدیل و اجرا می‌شوند. در صورت وجود JVM[۱] مانند سایر زبان‌های مبتنی بر آن که وابسته به سیستم‌عامل خاصی نیستند برنامه‌های نوشته شده به جاوا بر روی هر نوع سیستم عامل و هرگونه وسیله الکترونیکی قابل اجرا می‌باشند. شعار جاوا «یکبار بنویس و همه جا اجرا کن» است که به همین ویژگی اشاره دارد. این ویژگی جاوا را مستقل از سکو می‌نامند. [۲]

اسپرینگ یک چارچوب نرم‌افزاری و مدیر وارونگی کنترل متن‌باز برای سکوی جاوا است.

ویژگی‌های اصلی این فریمورک می‌تواند توسط هر برنامه جاوا مورد استفاده قرار گیرد، اما دارای افزونه‌هایی برای ساختن برنامه‌های کاربردی وب بر روی پلت فرم JavaEE[۲] می‌باشد. اگر چه این فریمورک هیچ مدل برنامه‌نویسی خاصی را به برنامه‌نویس تحمیل نمی‌کند، اما در میان برنامه نویسان جاوا به عنوان یک راهکار دوم، یا یک جایگزین یا حتی افزونه ای برای مدل Enterprise JavaBeans (EJB)تبدیل شده‌است.

این فریمورک توسط آقای راد جانسون (Rod Johnson) نوشته شده و برای اولین بار در سال ۲۰۰۳ ارائه شد. نسخه دوم در سال ۲۰۰۶ ارائه شده و موفق به دریافت جوایز متعدد گردید؛ و تاکنون تا نسخه ۴٫۳ از این فریمورک منتشر شده‌است.

[۱] Java virtual mechine
[۲] Java Enterprise edition

اسپرینگ یک فریمورک سبک است که می‌توان آن را فریمورک فریمورک‌ها نامید! به این خاطر که از انواع فریمورک‌ها دیگر پشتیبانی می‌کند. در نگاهی کلی تر اسپرینگ را به عنوان ساختاری تعریف می‌کنیم که در آن می‌توان solution های مربوط به مسایل تکنیکال مختلف را پیدا کرد. [۳]

هایبرنت یک کتابخانه نگاشت شی-رابطه ایobject-relational mapping برای زبان جاوا است که چارچوبی را برای نگاشت یک شی به یک پایگاه داده رابطه‌ای فراهم می‌آورد.
کاربرد اصلی هایبرنیت نگاشت یک کلاس جاوا به یک جدول در پایگاه داده است. هایبرنیت همچنین ابزاری را برای بازیابی داده‌ها فراهم می‌آورد. هایبرنیت کمک می‌کند یک برنامه‌نویس بدون استفاده مستقیم از دستورات مربوط به پایگاه داده با آن رابطه برقرار و شی‌ها را بازیابی، ذخیره یا به روزرسانی کند. [۴]
همچنین هایبرنت زبانی پرس و جو مخصوص به خود را دارد. همانند sql که از پایگاه داده اطلاعاتی را میخواند. اما sql خروجی جدول بر میگرداند و برنامه نویس باید اطلاعات جدول را به شی نگاشت کند ولی HQL یا hibernate query language پرس و جو بین کلاس ها می باشد و خروجی ان یک شی می باشد. در واقع هایبرنت شبیه سازی پایگاه داده های شی گرا می باشد [۴]

۲-۲- معماری MVC و Rest Full Api

معماری mvc از یک رویکرد سه لایه تبعیت می کند. هر بخش از معماری mvc دارای ویژگی های خاصی بوده و وظایف ویژه ای را به صورت مجزا بر عهده دارد.

لایه مدل

مدل، لایه‌ای از برنامه است که به ذخیره‌ی دائمی داده‌های برنامه مربوط است و به بیانی شفاف‌تر، با دیتابیس در ارتباط است. پردازش داده‌ها، چه پیش از ذخیره‌ی داده‌ها و چه پس از آن، هنگام استفاده، بر عهده‌ی لایه‌ی مدل است.
مهم است که بدانیم لایه‌ی مدل در این معماری، نسبت به آنچه در بیرون از آن اتفاق می‌افتد، نابینا است. این که داده‌هایی که ذخیره می‌شوند از کجا و به چه طریقی به دست آمده‌اند و این که داده‌هایی که استخراج می‌شوند قرار است به چه کار آیند، به لایه‌ی مدل ارتباطی ندارد. لایه‌ی مدل نه این چیزها را می‌داند و نه در طلب دانستن آن‌ها، ارتباطی با دیگر لایه‌ها برقرار می‌کند.
لایه‌ی مدل را نباید دیتابیس یا دروازه‌ی ارتباط با سامانه‌ای دیگر در نظر گرفت. لایه‌ی مدل، محلی است برای کار با داده‌ها، به معنای عام کلمه، بدون آن که از دلیل استفاده از داده‌ها بپرسد و اما و اگری بیاورد.
از آنجا که داده‌ها قلب هر نرم‌افزاری هستند، مدل‌های یک برنامه، اغلب پیچیده‌ترین قسمت هر برنامه قلمداد می‌شوند که فرآیندهای اصلی در آنجا روی می‌دهد.

لایه نمایش

درک لایه‌ی نمایش، از همه ساده‌تر است. این لایه همان است که رابط کاربری را رقم می‌زند و ابتدا و انتهای چرخه‌ی برنامه است.
در واقع کاربران از این لایه با برنامه ارتباط برقرار می‌کنند و داده‌هایی را وارد می‌کنند و سپس در همین لایه نتیجه را مشاهده می‌نمایند.
در نرم‌افزارهای تحت وب لایه‌ی نمایش همانتگ های HTML هستند که با CSS زیبا میشوند.

لایه کنترلر

لایه‌ی کنترل، رگ حیاتی برنامه است که اجزای آن را به یکدیگر متصل می‌سازد. درخواستی که کاربران از طریق لایه‌ی نمایش صادر کرده‌اند، در لایه‌ی کنترل وصول می‌شود، به مسیر درست هدایت می‌شود، اعتبارسنجی‌های لازم (به کمک لایه‌ی مدل) روی آن صورت می‌گیرد، و اگر مشکلی نباشد نهایتاً برای ذخیره‌سازی یا استخراج به لایه‌ی مدل ارسال می‌شود و نتیجه‌ها یا بازخوردهای آن دوباره به لایه‌ی نمایش بازگردانده می‌شود تا به اطلاع کاربر برسد.

باید توجه داشت که در یک برنامه‌ی تحت وب، همه‌ی درخواست‌های کاربران لزوماً از لایه‌ی نمایش عبور نمی‌کنند. اگر از APIها که اساساً لایه‌ای به عنوان لایه‌ی نمایش ندارند نیز بگذریم، بسیاری اوقات شما با کلیک بر لینکی که از قبل و توسط شخصی دیگر مهیا شده وارد برنامه می‌شوید و به این صورت، بدون عبور از لایه‌ی نمایش، درخواست خود را مستقیماً به لایه‌ی کنترل می‌فرستید. [۷]

یکی از بخش‌های لاینفک وب مدرن، ای پی آی ها، به‌کارگیری از آن‌ها، توسعه ی آن‌ها و مهم از همه معماری RESTful API است که هر توسعه‌دهنده ی وب اپلیکیشنی، باید با سازوکار آن آشنایی داشته باشد که در این آموزش قصد داریم نگاهی کلی به مفهوم رستفول ای پی آی داشته باشیم.

RESTمخفف واژگان Representational State Transfer است که از سال ۲۰۰۵ در وب شناخته شد که در ظاهر کمی گیج‌کننده به نظر می‌رسد، اما با کمی توضیح، می‌توان این مفهوم در ظاهر پیچیده را رمزگشایی کرد!

اگر خیلی ساده بخواهیم به این قضیه نگاه کنیم، RESTعبارت است از راه کارها و روش‌هایی که با استفاده از آن‌ها می‌توان به رد و بدل دیتا از طریق شبکه پرداخت. به عبارت دیگر، RESTراهی ساده به منظور سازماندهی تعاملات مابین سیستم‌های مجزا از یکدیگر می‌باشد.

در مقابلREST ، پروتکل SOAP که مخفف واژگان Simple Object Access Protocol است قرار دارد که از طریق آن می‌توان به رد و بدل دیتا از طریق شبکه در قالب وب سرویس های مختلفی با فرمت XML پرداخت.

APIهم مخفف واژگان Application Programming Interface است که دربرگیرنده ی متدهایی برای ارتباط با سایر لایبرری ها یا اپلیکیشن ها است .
حال اگر این اصطلاحات در کنار یکدیگر قرار دهیم و چیزی تحت عنوان RESTful API بسازیم، منظورمان سازوکارهایی برای ارتباط با سایر سرویس ها با استفاده از معماری خاصی است.

معماری REST دارای یکسری ویژگی‌ها است که شاخص ترین آن‌ها عبارتند از:
-ثبات و یکنواختی این معماری در جای جایAPI
-عدم برخورداری از سشن در سمت سرور
-به‌کارگیری از کدهای وضعیت اچ تی تی پی
-استفاده از یو آر ال ها برای مشخص ساختن مسیرهای مد نظر
- اِعمال کوئری ها در یو آر ال به جای هدر پروتکل اچ تی تی پی [۵]
برای به‌کارگیری ازRESTful API ، چهار متد پیش رو داریم که عبارتند از:
GET: -برای دریافت یک آبجت
POST: -برای ساخت و ارسال یک آبجکت
PUT: -برای تغییر و جایگزین کردن یک آبجکت
- DELETE:برای حذف یک آبجکت

۲-۳- react js و jwt

ری اکت یک کتابخانه متن‌باز جاوااسکریپت برای ساخت رابط‌های کاربری و اجزای Component صفحات وب است.
این کتابخانه توسط فیس‌بوک و جامعه‌ای از توسعه‌دهندگان و شرکت‌ها به صورت انفرادی توسعه و نگه‌داری می‌شوند.[۲] براساس آنالیزهای جاوااسکریپت سرویسLibescore ، ری‌اکت در حال حاضر در سایت‌های نت‌فلیکس، Imgur، بلیچر رپورت، فیدلی، ایر بی‌ان‌بی و ... مورد استفاده قرار می‌گیرد.
به دلیل بهینه بودن ری‌اکت برای دریافت اطلاعاتی که با سرعت تغیر میکنند، میتوان از آن برای توسعه برنامه تک‌صفحه‌ای SPA یا برنامه‌های موبایل استفاده کرد. هرچند دریافت اطلاعات ابتدایی ترین بخش در یک صفحه وب است و برنامه‌های پیچیده ری‌اکت معمولا به کتابخانه‌های اضافه‌ای برای مدیریت وضعیت state management ، مسیریابی URL mapping ، و اتصال به رابط برنامه‌نویسی کاربردیAPI نیاز دارند.
ویژگی های ری اکت:
DOM مجازی
وقتی صفحه وبی بارگذاری میشود مرورگر یک مدل شی گرا از المان های موجود در صفحه می سازد.این مدل Dom و یا Document Object Model نام دارد. این نمودار یک نمودار درختی از اشیا موجود در صفحه است. برنامه هایی مانند جاوااسکریپت با استفاده از این مدل یک صفحه Html را به صورت پویا ایجاد می کنند.

در تکنولوژیReact.js از مفهومی به نام Dom مجازی (Virtual DOM) استفاده می شود و به این صورت کار میکند که برای هر شی Dom که ویژگی جدیدی قرار است برای آن ساخته شود نیازی نیست که آن شی دوباره ساخته شود بلکه تنها نیاز است که آن ویژگی مورد نظر را تغییر داد و یا افزود.

جی‌اس‌ایکس(JSX)
جی‌اس‌ایکس یک نسخه گسترش یافته از جاوااسکریپت است که این امکان را می‌دهد تا بتوان در کنار کدهای جاوااسکریپت از کدهای اچ‌تی‌ام‌ال نیز بهره برد که به موجب آن کامپوننت‌های ری‌اکت معمولاً در قالب جی‌اس‌ایکس نوشته می‌شوند. همچنین ممکن است توسعه دهندگان تنها از جاوااسکریپت خالص استفاده کنند. [۶]

JWT یا JSON Web Token

دو روش کلی و پرکاربرد اعتبارسنجی سمت سرور، برای برنامه‌های سمت کاربر وب وجود دارند:

روش اول Cookie-Based Authentication که پرکاربردترین روش بوده و در این حالت به ازای هر درخواست، یک کوکی جهت اعتبارسنجی کاربر به سمت سرور ارسال می‌شود.
روش دوم Token-Based Authentication که بر مبنای ارسال یک توکن امضا شده به سرور، به ازای هر درخواست است.

JWT یا JSON Web Token چیست؟

توضیح JSON Web Token :یا JWT یک استاندارد وب است RFC ۷۵۱۹ که روشی فشرده و خود شمول (self-contained) را جهت انتقال امن اطلاعات بین مقاصد مختلف را توسط یک شی JSON تعریف می‌کند. این اطلاعات قابل تصدیق و اطمینان هستند، از این رو که به صورت دیجیتال امضا می‌شوند. JWT ها توسط یک کلید خصوصی با استفاده از الگوریتم HMAC و یا یک جفت کلید خصوصی و عمومی توسط الگوریتم RSA قابل امضا شدن هستند.
-فشرده بودن: اندازه‌ی شی JSON یک توکن در این حالت کوچک بوده و به سادگی از طریق یک URL و یا پارامترهای POST و یا داخل یک HTTP Header قابل ارسال است و به دلیل کوچک بودن این اندازه انتقال آن نیز سریع است.
-خودشمول: بار مفید (payload) این توکن شامل تمام اطلاعات مورد نیاز جهت اعتبارسنجی یک کاربر است تا دیگر نیازی به کوئری گرفتن هر باره از بانک اطلاعاتی نباشد در این روش مرسوم است که فقط یکبار از بانک اطلاعاتی کوئری گرفته شده و اطلاعات مرتبط با کاربر را امضای دیجیتال کرده و به سمت کاربر ارسال می‌کنند.

جوت‌ها دارای سه قسمت است و این قسمت‌ها با نقطه از هم جا شدند، مانند: xxxxx.yyyyy.zzzzz که شامل سه بخش: header.payload.signature می‌باشند و هر بخش با الگوریتم Base۶۴ اینکد می‌شود.

بخش:Header

بخش Header عموما دارای دو قسمت است که نوع توکن و الگوریتم مورد استفاده توسط آن را مشخص می‌کند. نوع توکن در اینجا JWT است و الگوریتم‌های مورد استفاده، عموما HMAC SHA۲۵۶ و یا RSA هستند

بخش Payload

بخش payload یا «بار مفید» توکن، شامل claims است، منظور از claims اطلاعاتی است در مورد موجودیت مدنظر (عموما کاربر) و

یک سری متادیتای اضافی. سه نوع claim وجود دارند:

نوع اول :Reserved claims یک سری اطلاعات مفید و از پیش تعیین شده غیراجباری هستند مانند iss:یا صادر کنند، expیا تاریخ انقضا، subیا عنوان (subject) و aud یا مخاطب(audience)

نوع دوم :Public claims می‌تواند شامل اطلاعاتی باشد که توسط IANA JSON Web Token Registry پیشتر ثبت شده‌است و فضاهای نام آنها تداخلی نداشته باشند.

نوع سوم :Private claimsادعای سفارشی هستند که جهت انتقال داده‌ها بین مقاصد مختلف مورد استفاده قرار می‌گیرند.

بخش signature

بخش اول (هدر) با بخش دوم (پیلود) جمع شده و بعد با کلید خصوصی (کلید خصوصی فقط در سرور موجوده) رمز میشه، این به عنوان بخش سوم یا امضا توکن ما استفاده میشه، درصورتی که اطاعات پیلود دست کاری بشه امضا برای سرور معتبر نیست، همچنین چون کاربران کلید خصوصی ندارند نمی‌تونند خودشون توکن تولید کنند [۷]

فصل سوم – تحلیل و طراحی سیستم

۳-۱- تحلیل سیستم نوبت دهی پزشکی

شرح فرایند ها (Bussiness Process)

۱- **ثبت اطلاعات بیمار**:

بیمار در اولین درخواست خود اطلاعات خود را باید به منشی بدهد تا در دفتر بیماران ثبت کند این اطلاعات شامل:
نام، نام خانوادگی ، کد ملی، شماره تلفن و جنیست می باشد

۲- **درخواست نوبت توسط بیمار** :

بیمار با مطب دکتر تماس گرفته و از منشی درخواست نوبت میکند منشی در اولین زمان خالی که دکتر میتواند ویزیت کند را برای بیمار مشخص کرده و در دفتر نوبت ها شماره بیمار و زمان نوبت او را ثبت میکند
اطلاعاتی که از بیمار گرفته میشود شامل:
نام و نام خانوادگی، نوع مراجعه (معاینه ، پر کردن دندان، کشیدن دندان، ...)

۳- **تکمیل نوبت توسط بیمار**:

زمانی که نوبت بیمار فرا میرسد اگر به کلینک مراجعه کند با دادن نام و نام خانوادگی خود در صف نشسته تا دکتر او را معاینه کند
بعد از معاینه دکتر نوبت منشی در دفتر نوبت ثبت میکند که بیمار مراجعه کرده و از نوبت خود استفاده کرده است
و بعد از معاینه دکتر منشی با توجه به نوع مراجعه بیمار هزینه ای را از او میگیرد

۴- **لغو نوبت توسط بیمار**:

اگر بیمار به هر دلیل نتواند در زمان مشخص شده برای او به دکتر مراجعه کند باید با منشی تماس گرفته و درخواست لغو نوبت دهد

منشی نیز در دفتر نوبت ها نام اورا حذف میکند و ان ساعت را برای بقیه مراجعه کنندگان در نظر میگیرد

اطلاعاتی که کاربر باید ارائه دهد:

نام و نام خانوادگی و تاریخ نوبت

۵- **لغو نوبت توسط دکتر**

دکتر ممکن است در زمانی که به بیماری نوبت داده نتواند او را معاینه کند

منشی باید بیمار تماس گرفته و تاریخ جدید برای او مشخص کند

۶- **تغییر تاریخ نوبت**

بنا به درخواست پزشک یا بیمار تاریخ نوبت میتواند به تاریخی دیگر تغیر پیدا کند

۷- **گزارشی از هر بیمار**

هر بیمار هر نوبتی که از پزشک میگیرد در سیستم او ثبت میشود و پزشک میتواند تاریخچه مراجعه ان بیمار به مطب خود را ببیند

۸- **گرفتن نوبت معاینه**

بیمار میتواند از دکتر نوبت معاینه بگیرد نوبت معاینه به دلیل انیکه سریع انجام میشود معمولا تاریخ مشخصی ندارند و بنا به درخواست بیمار تعیین می شوند

شناسه های موجود در سیتم (Bussiness Object)

۱- بیمار

۲- پزشک (منشی)

۳- اطلاعات نوبت

۴- اطلاعات بیماری

۳-۲- سند نیازمندی ها

نیازمندی های غیرعملیاتی (NoneFunctional Requirment)

طراحی رابط کاربری مناسب

ایجاد سه نوع نقش در سیستم و تعیین سطح دسترسی های هر نقش

احراز هویت و مجوز با استفاده از JWT

ساخت سیستم full rest api

ایجاد سیستم با معماری MVC

حفاظت از فایل های اپلود شده بر روی سرور

نیازمندی های عملیاتی (Functional Requirment)

ثبت نام در سیستم (پزشک، بیمار، منشی)

خروج از حساب کاربری

ورود به حساب کاربری

فراموشی رمز عبور

مشاهده و در صورت نیاز لغو نوبت های گرفته شده برای بیمار

جست و جوی پزشک با نام ، شهر و تخصص

رزرو نوبت توسط پزشک

پروفایل اختصاص برای هر کاربر (پزشک ، بیمار)

تغییر مشخصات پروفایل

نمایش نوبت های پزشک

تغییر یا لغو نوبت توسط پزشک

در این صورت باید در پروفایل دکتر پیامی مبتی بر لغو شدن یا تغییر نوبت بیمار داده شود

تعیین نوع سرویس هایی که پزشک ارائه میدهد

هر پزشک با توجه به تخصص خود سرویس هایی را ارائه می دهد برای مثال پزشک دندان پزشک سرویس هایی از قبیل پرکردن، عصب کشی و... را ارائه میدهد

تعیین نوع بیمه های طرف قرارداد

اضافه کردن بیمار جدید به سیستم توسط پزشک

جستجو بیمارانی که به پزشک مراجعه کردند

گزارش گیری در بازه زمانی مشخص شده از نوبت ها و درامد پزشک در این بازه

حذف و اضافه کردن پزشک به کلینک

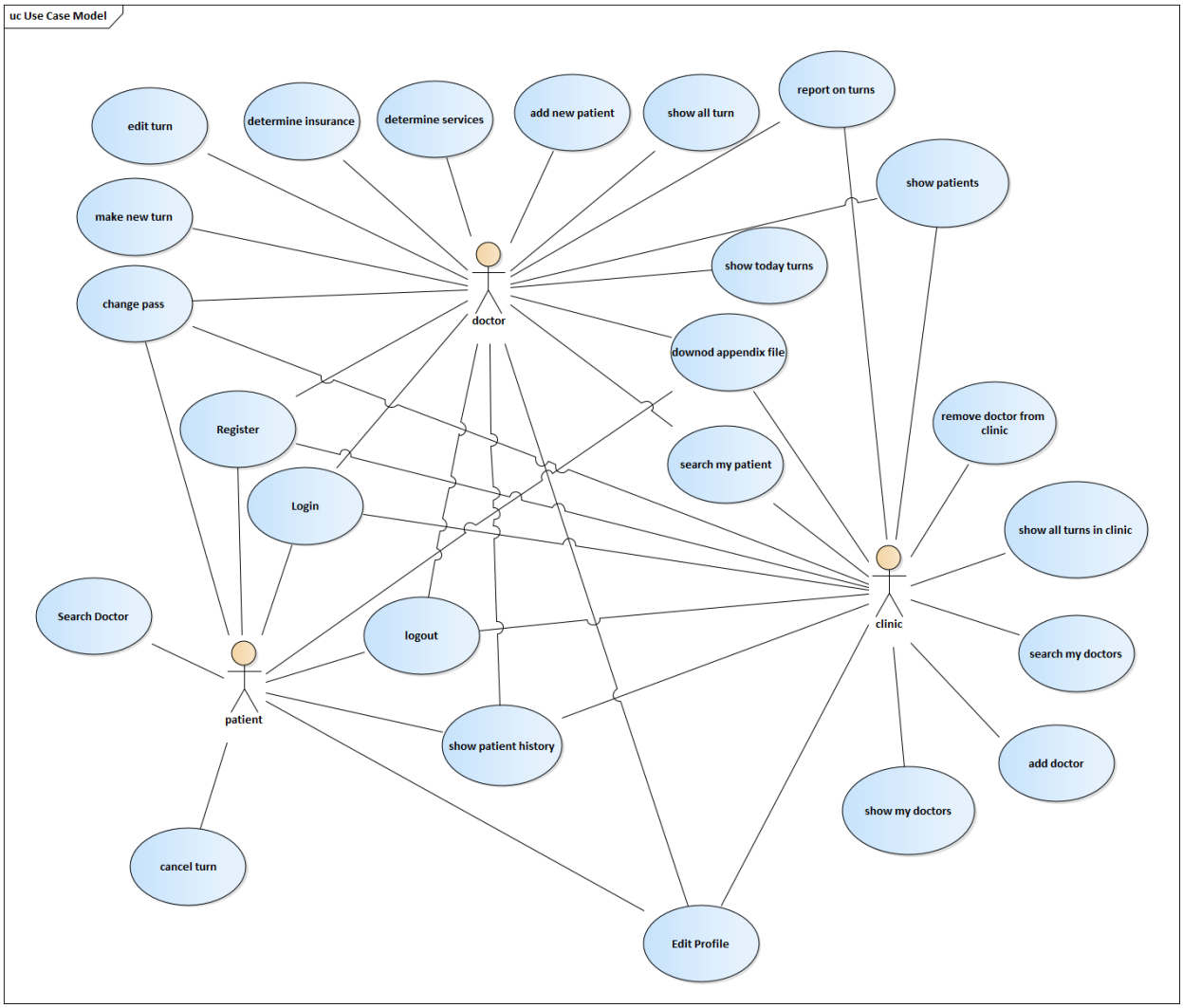
جستجو پزشکان کلینیک توسط ادمین کلینیک

گزارش گیری در بازه زمانی مشخص از نوبت ها و درامد پزشکان کلینیک

دانلود فایل های پیوست شده

۳-۳- دیاگرام

دیاگرام مورد کاربرد (use case diagram)



شکل ۳-۱ دیاگرام مورد کاربرد

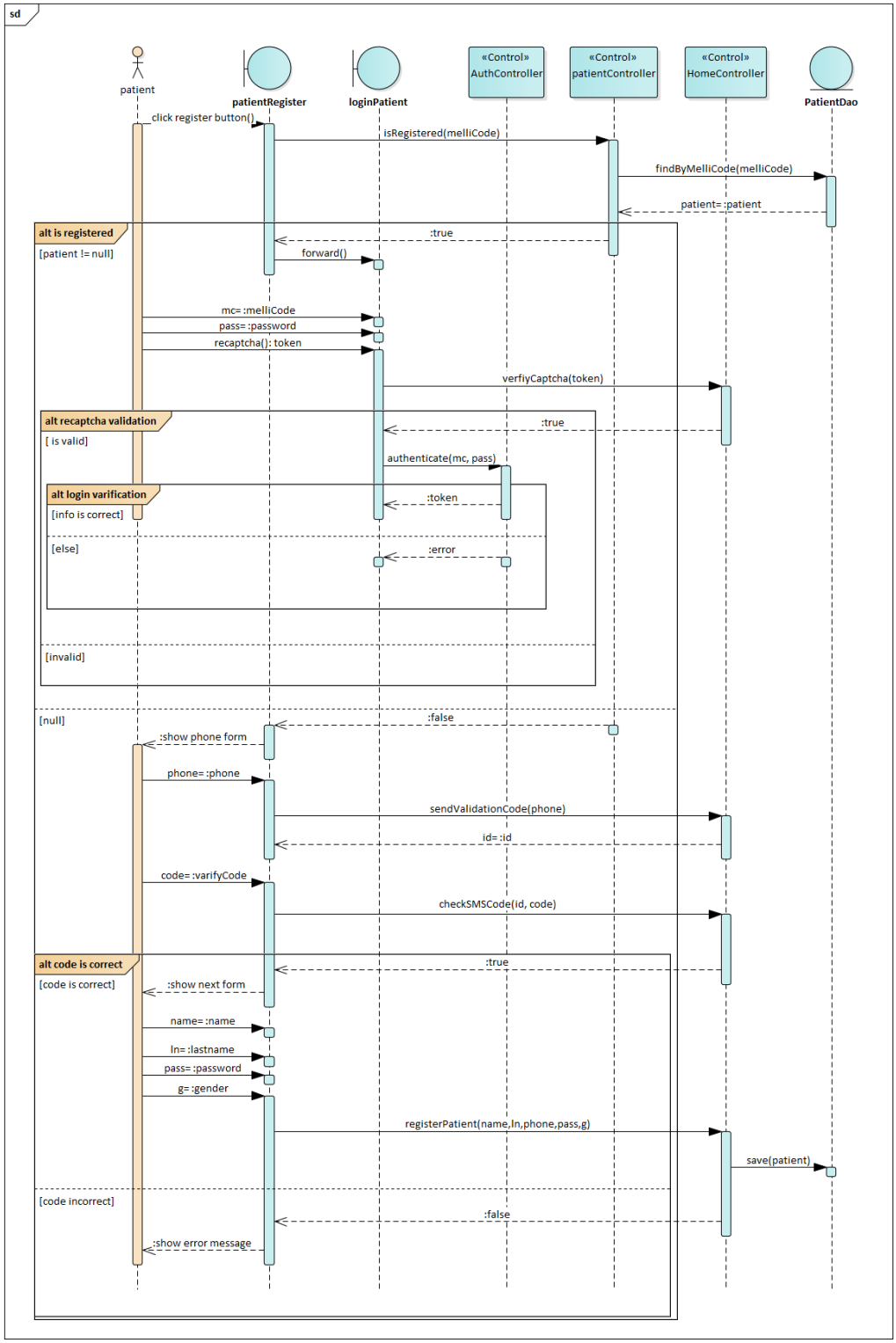
شرح دیاگرام مورد کاربرد به همراه نمودار توالی و فعالیت

جدول ۳-۱ مورد کاربرد ثبت نام بیمار

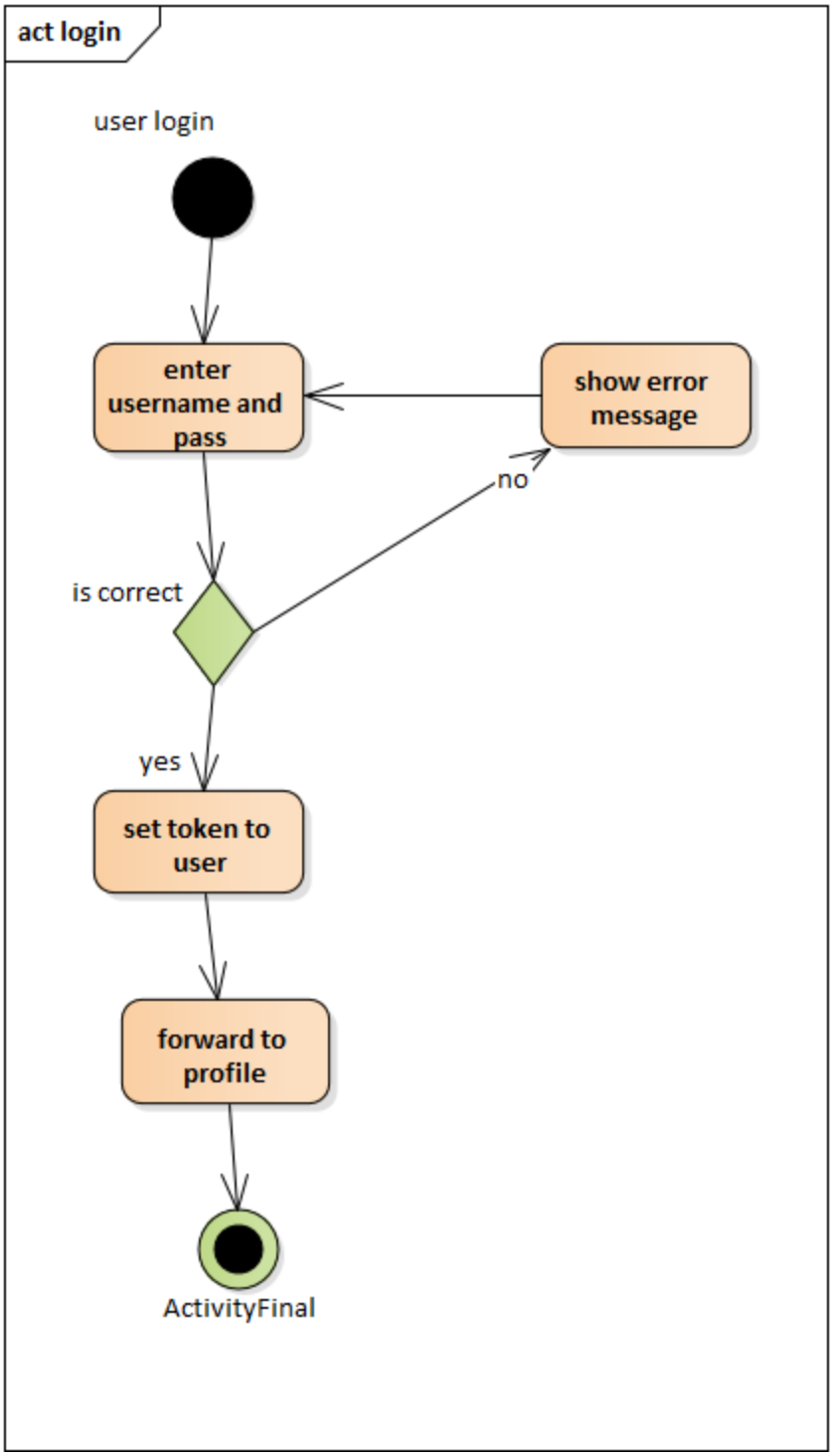
Use Case Number	۱
Use Case	ثبت نام بیمار
Summary	هر بیمار باید در سیستم ثبت نام کند تا پزشکان بتوانند برای او نوبت ثبت کنند
Actor	بیمار
Trigger	دکمه ورود/عضویت
Primary Scenario	کاربر کد ملی خود را وارد میکند اگر قبلا در سیستم ثبت نام نکرده باشد باید شماره تلفن خودش را وارد و تایید کند پیامک حاوی کد تایید برای او ارسال میشود بعد از تایید نام، نام خانوادگی، رمز، و جنسیت را وارد میکند
Alternative Scenario	پزشک مستقیما برای بیمار حساب کاربری ایجادکند
Exceptional Scenario	۱- اگر اطلاعات وارد شده با فرمت تعیین شده برابر نباشد اطلاعات ثبت نشده و پیام خطا به او نمایش داده میشود ۲- اطلاعاتی که به عنوان ضروری تعیین شده باید حتما پر شوند در غیر این صورت پیام خطا مربوطه نمایش داده می شود ۳- بیمار با کد ملی که در سیستم ثبت شده نمی تواند ثبت نام کند ۴- بیمار با شماره ای که قبلا در سیستم ثبت شده نمی تواند ثبت نام کند
Pre-Conditions	۱- بیمار برای ثبت نام نیاز به یک شماره تلفن دارد ۲- بیمار باید کد ملی معتبر داشته باشد

جدول ۳-۱ مورد کاربرد ورود بیمار

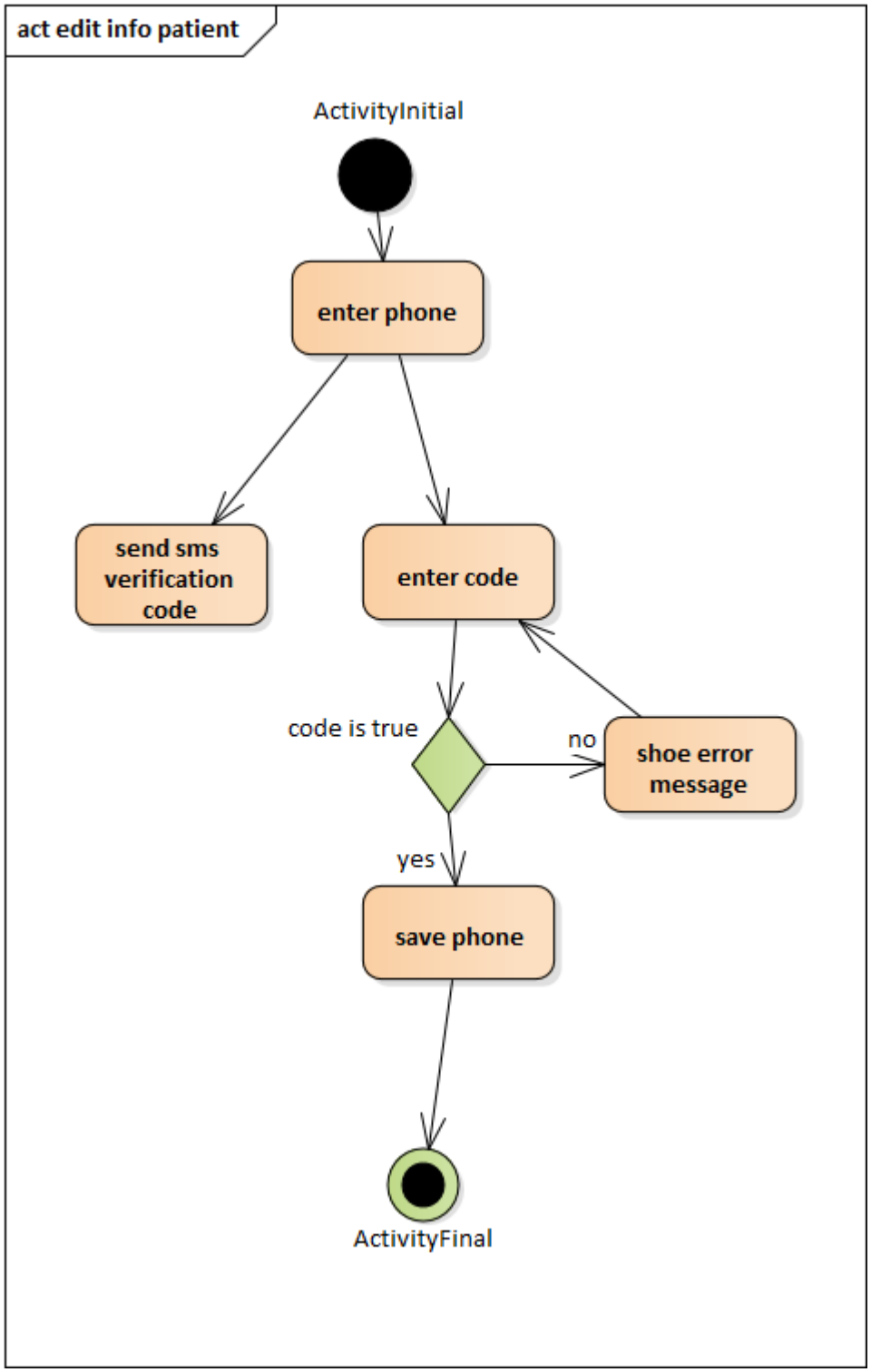
Use Case Number	۲
Use Case	ورود بیمار
Summary	بیمار برای مشاهده نوبت های خود باید وارد سیستم شود
Actor	بیمار
Trigger	دکمه ورود/عضویت
Primary Scenario	بیمار کدملی خود را وارد میکند, سپس رمز عبور خود را وارد و ریکپچا را تایید میکند
Exceptional Scenario	۱- اگر بیمار قبلا ثبت نام نکرده باشد فرم ثبت نام برای او نمایش داده می شود ۲- اگر رمز عبور نادرست باشد اجازه ورود داده نمی شود ۳- ربات اجازه ورود به سایت را ندارد
Pre-Conditions	بیمار قبلا ثبت نام کرده باشد
Post-Conditions	توکنی با نقش بیمار به کاربر داده میشود که میتواند به جاهایی که بیمار دسترسی دارد وارد شود



شکل ۳-۲ نمودار توالی ورود و ثبت نام بیمار



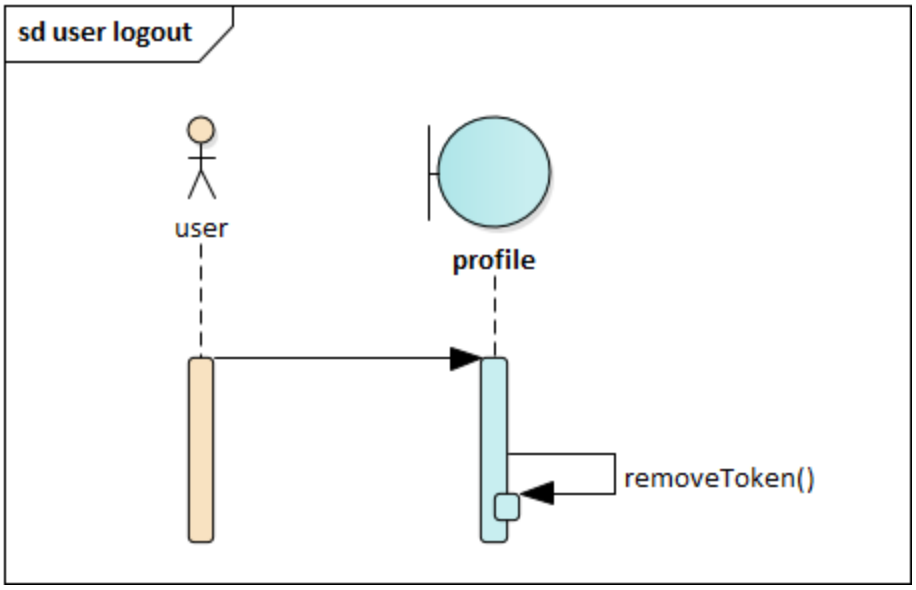
شکل ۳-۳ نمودار فعالیت ورود



شکل ۳-۴ نمودار فعالیت ثبت نام

جدول ۳-۳ مورد کاربرد خروج

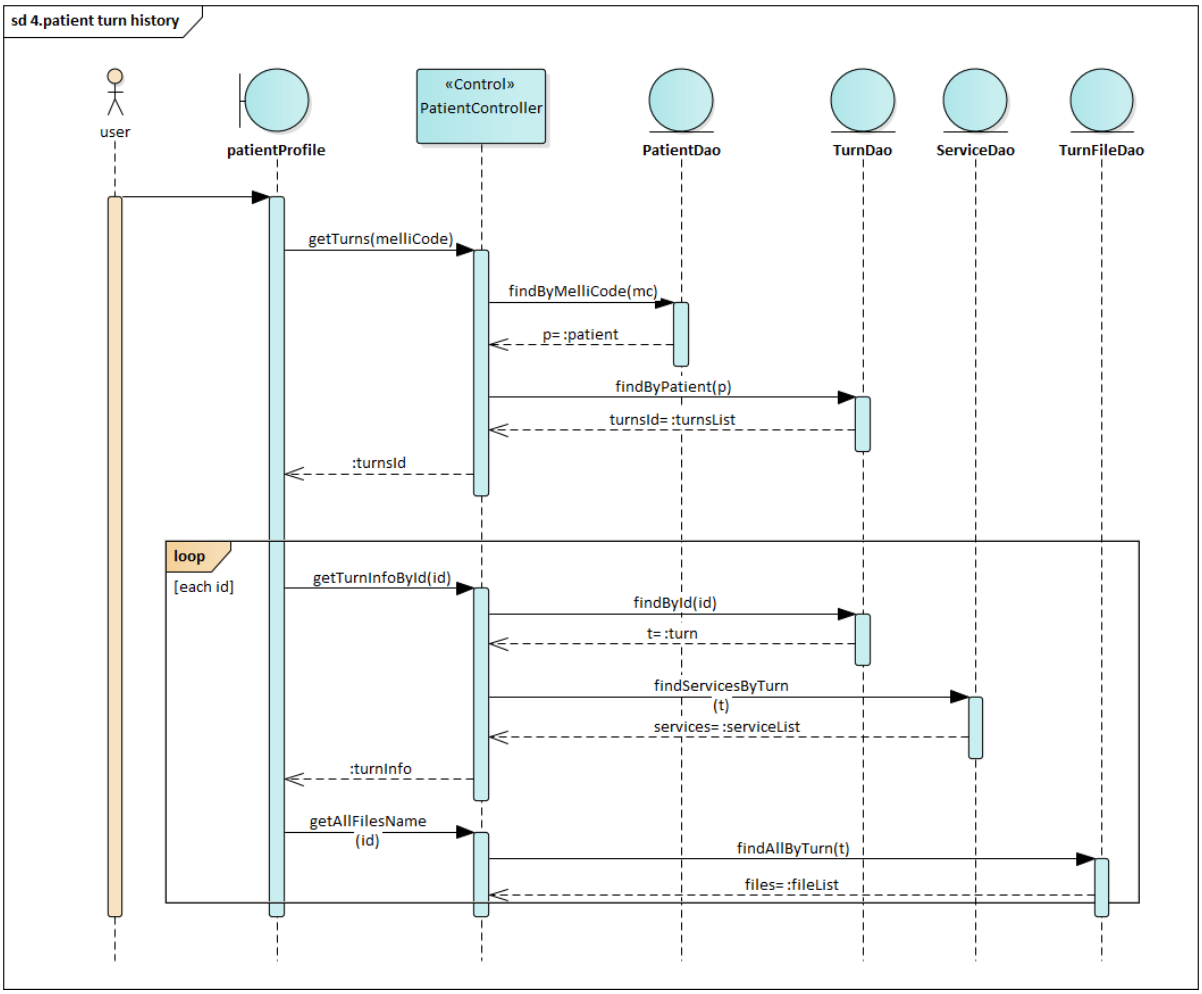
Use Case Number	۳
Use Case	خروج
Summary	کاربر میتواند قبل از خارج شدن از نرم افزار از حساب خود خارج شود
Actor	بیمار، پزشک و کلینیک
Trigger	دکمه خروج
Primary Scenario	کلیک روی دکمه خروج
Pre-Conditions	۱- کاربر باید به حساب کاربری خود وارد شده باشد
Post-Conditions	با خارج شدن بیمار توکن او ازبین میرود



شکل ۳-۵ نمودار توالی خروج

جدول ۳-۴ نمایش سابقه بیمار

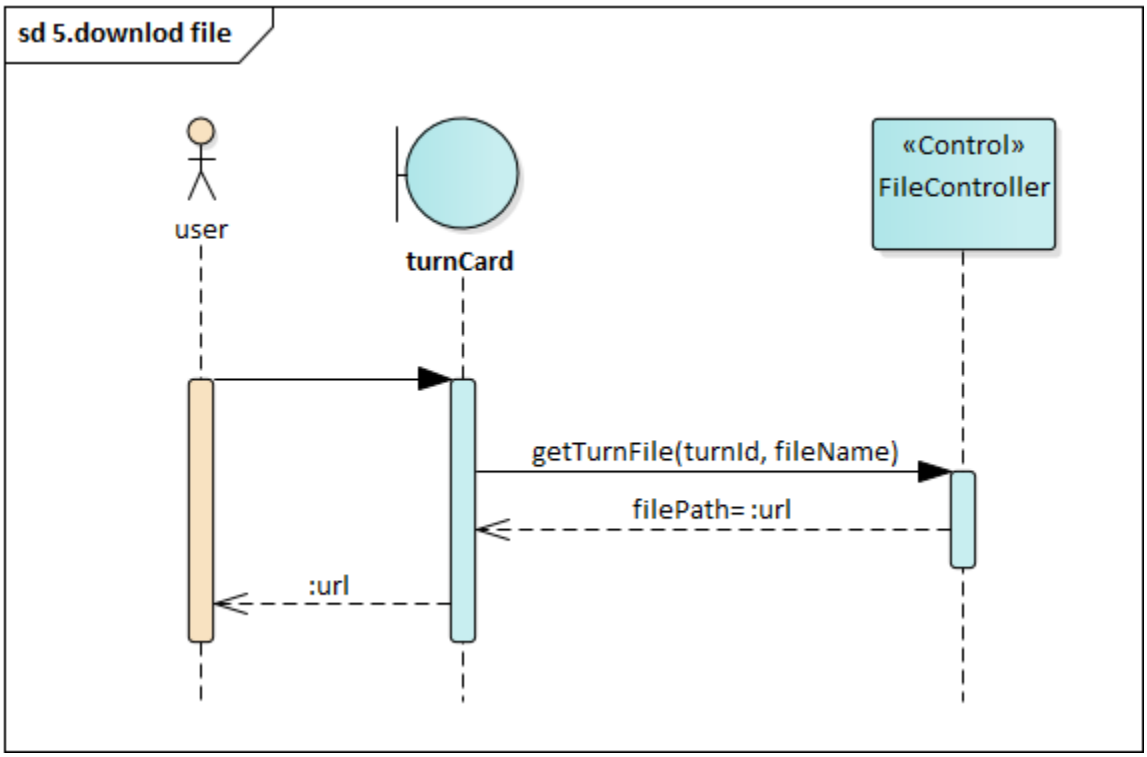
Use Case Number	۴
Use Case	نمایش تاریخچه نوبت های بیمار
Summary	کاربر میتواند نوبت هایی که برای بیمار گرفته شده است را ببیند
Actor	بیمار، پزشک، کلینیک
Trigger	دکمه نوبت های من برای بیمار و جزئیات در بیماران من برای پزشک و کلینیک
Primary Scenario	لیست از نوبت های که برای بیمار گرفته شده نمایش داده میشود این نوبت ها شامل نوبت های درحال انتظار(هنوز بیمار به پزشک مراجعه نکرده)، ویزیت شده(بیمار توسط پزشک ویزیت شده) و لغو شده می باشد. هر نوبت میتواند فایل هایی از قبیل تصویر، pdf و... به ان پیوست شده باشد که برای هر نوبت نمایش داده شده و بیمار با کلیک بر روی هر یک میتواند ان فایل را دانلود کند
Pre-Conditions	بیمار باید به حساب کاربری خود وارد شده باشد



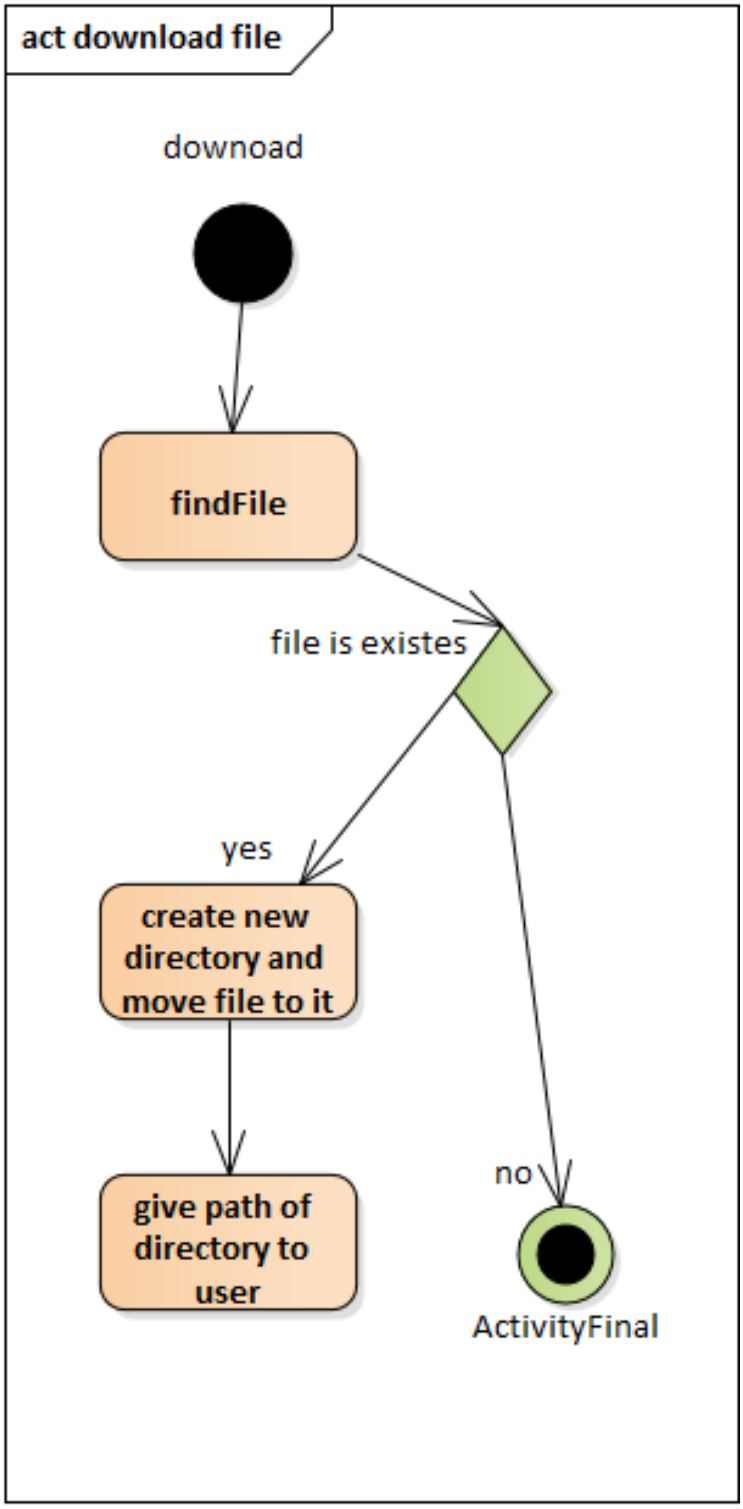
شکل ۳-۶ نمودار توالی نمایش سابقه بیمار

جدول ۳-۵ مورد کاربرد دانلود فایل

Use Case Number	۵
Use Case	دانلود فایل های پیوست شده
Summary	کاربر میتواند فایل های پیوست شده به نوبت ها را دانلود کند
Actor	بیمار، کلینیک، پزشک
Trigger	ایکن فایل
Primary Scenario	فایل در یک URL رندم برای کاربر قرار داده می شود و پس از ۳۰ دقیقه ان URL نامعتبر می شود
Pre-Conditions	کاربر باید به حساب کاربری خود وارد شده باشد



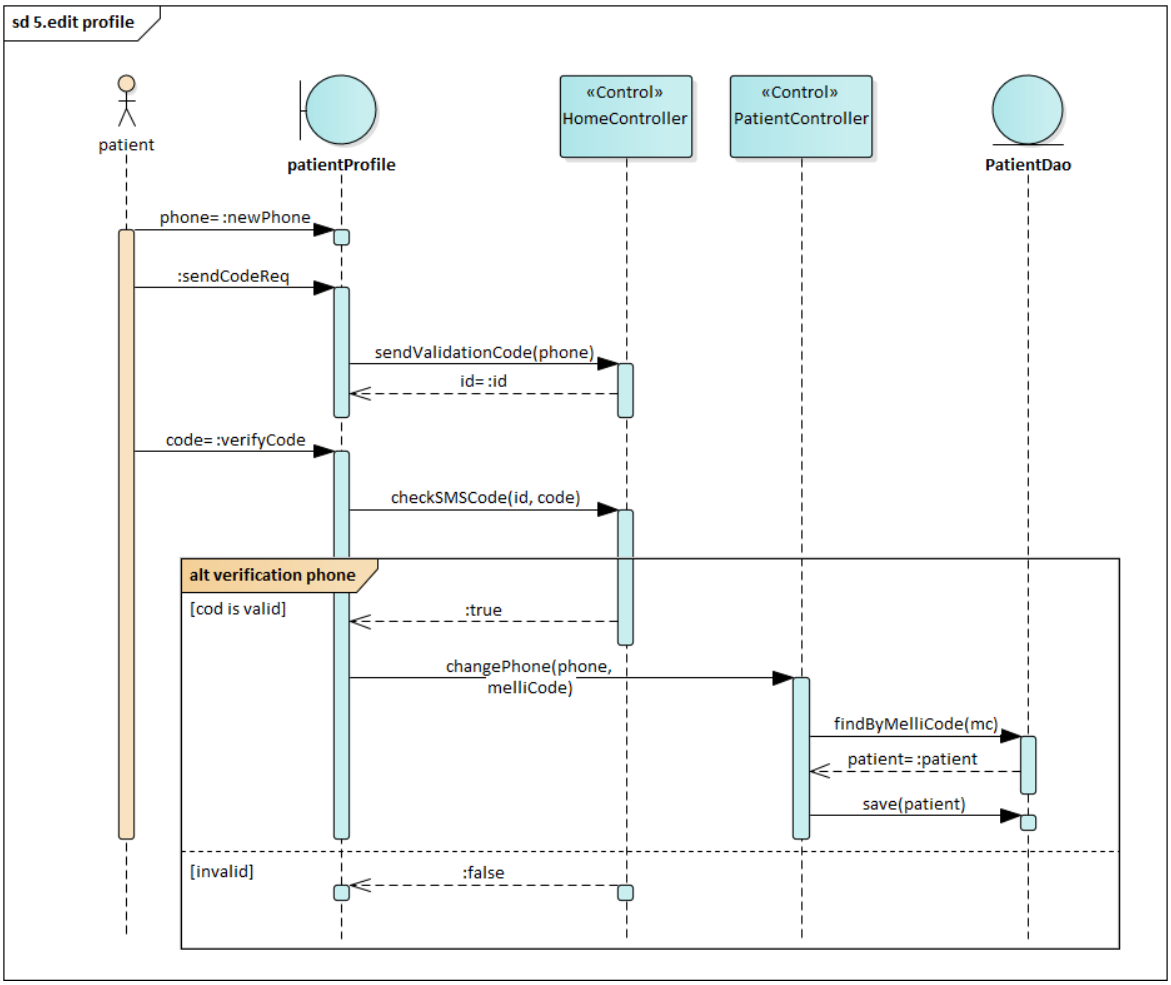
شکل ۳-۷ نمودار توالی دانلود فایل



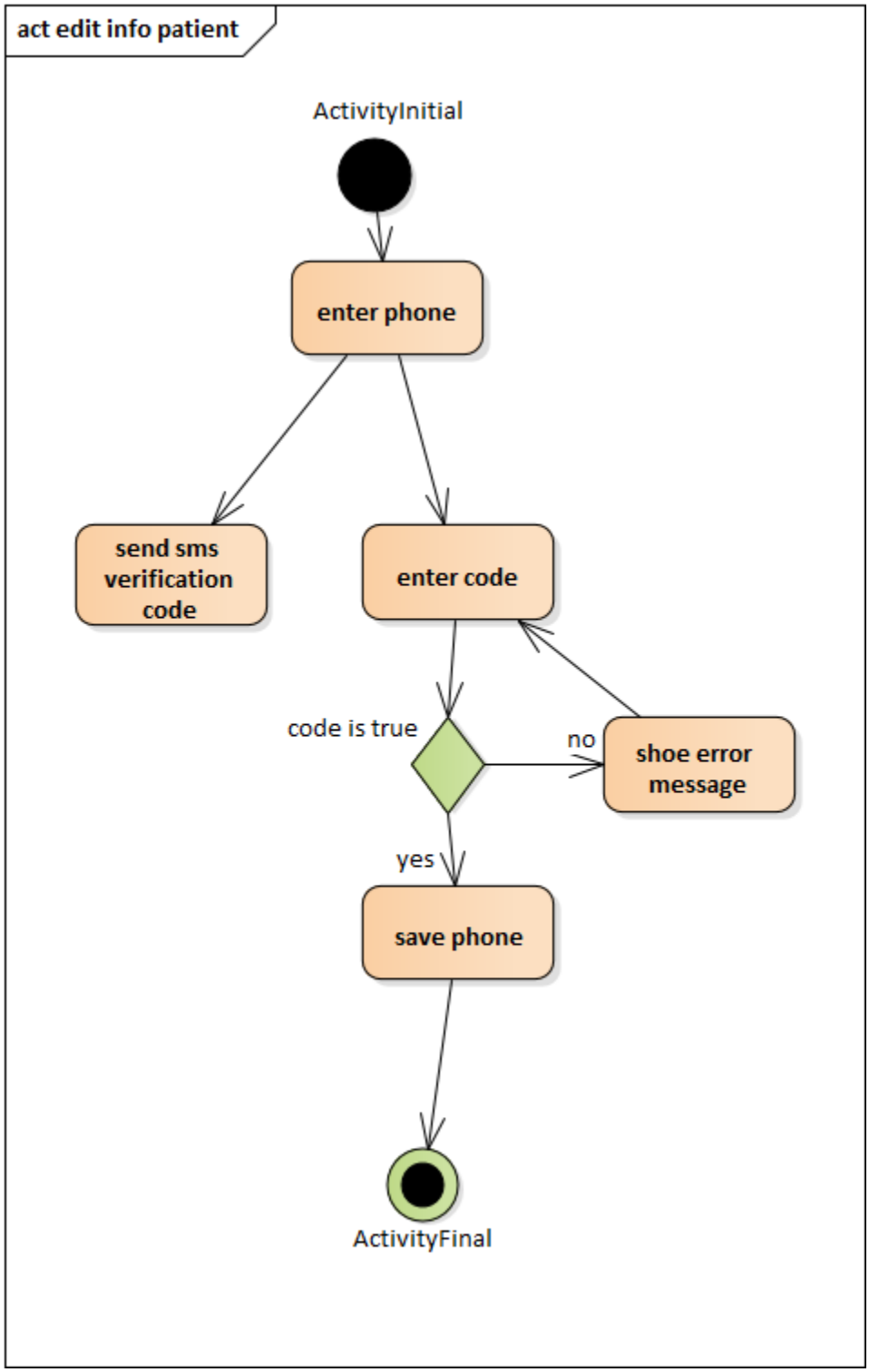
شکل ۳-۸ نمودار فعالیت دانلود فایل

جدول ۳-۶ مورد کاربرد تغییر مشخصات بیمار

Use Case Number	۶
Use Case	تغیر مشخصات کاربری
Summary	بیمار میتواند مشخصات خود را تغییر دهد
Actor	بیمار
Trigger	دکمه ویرایش
Primary Scenario	بیمار از مشخصات کاربری خود فقط قادر است تلفن خود را تغییر دهد بعد از کلیک بر روی دکمه ویرایش شماره جدید خود را وارد میکند پیامک حاوی کد تایید برای او ارسال میشود بعد از وارد کردن کد تایید در صورت درستی کد شماره تغییر میکند
Exceptional Scenario	۱- اگر کد تایید نادرست باشد پیام مناسب برای او نمایش و اجازه تغییر شماره را نمی دهد
Pre-Conditions	داشتن شماره تلفن معتبر



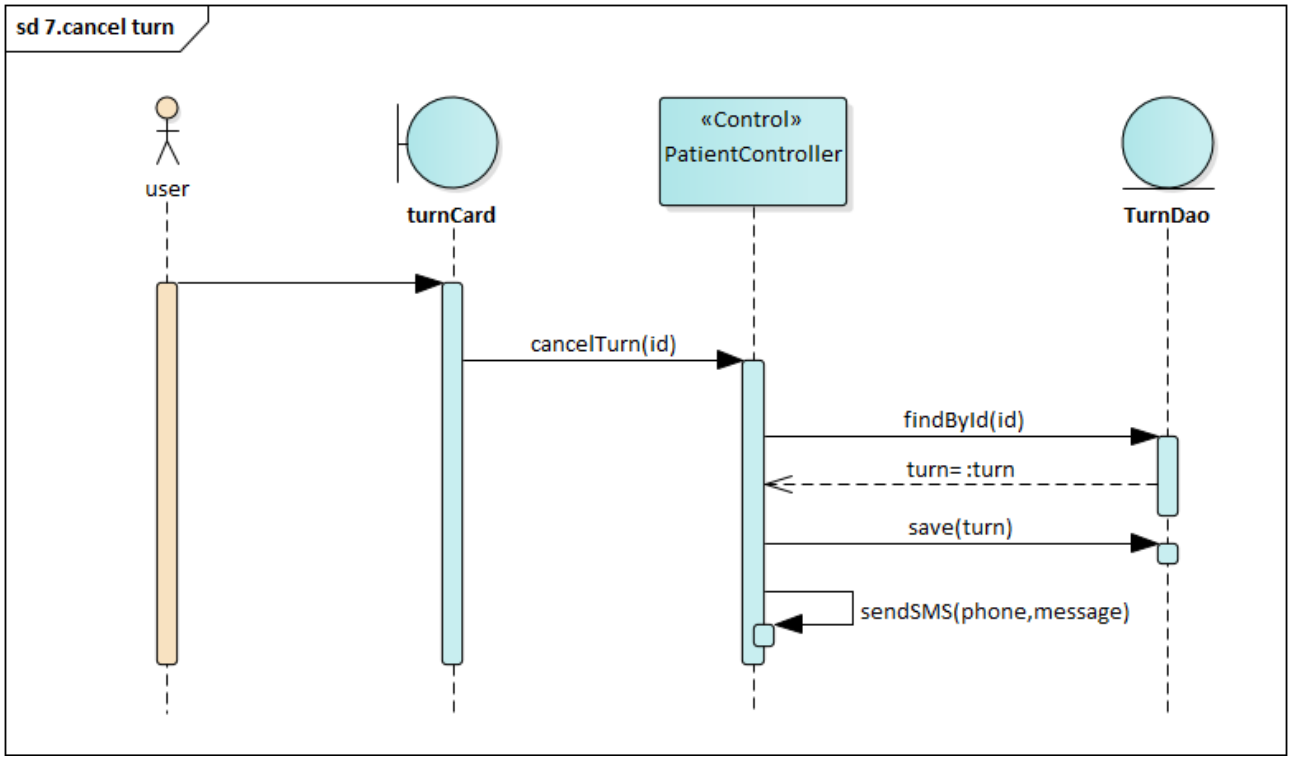
شکل ۳-۸ نمودار توالی تغییر مشخصات بیمار



شکل ۳-۹ نمودار فعالیت تغییر مشخصات بیمار

جدول ۳-۷ مورد کاربرد لغو نوبت

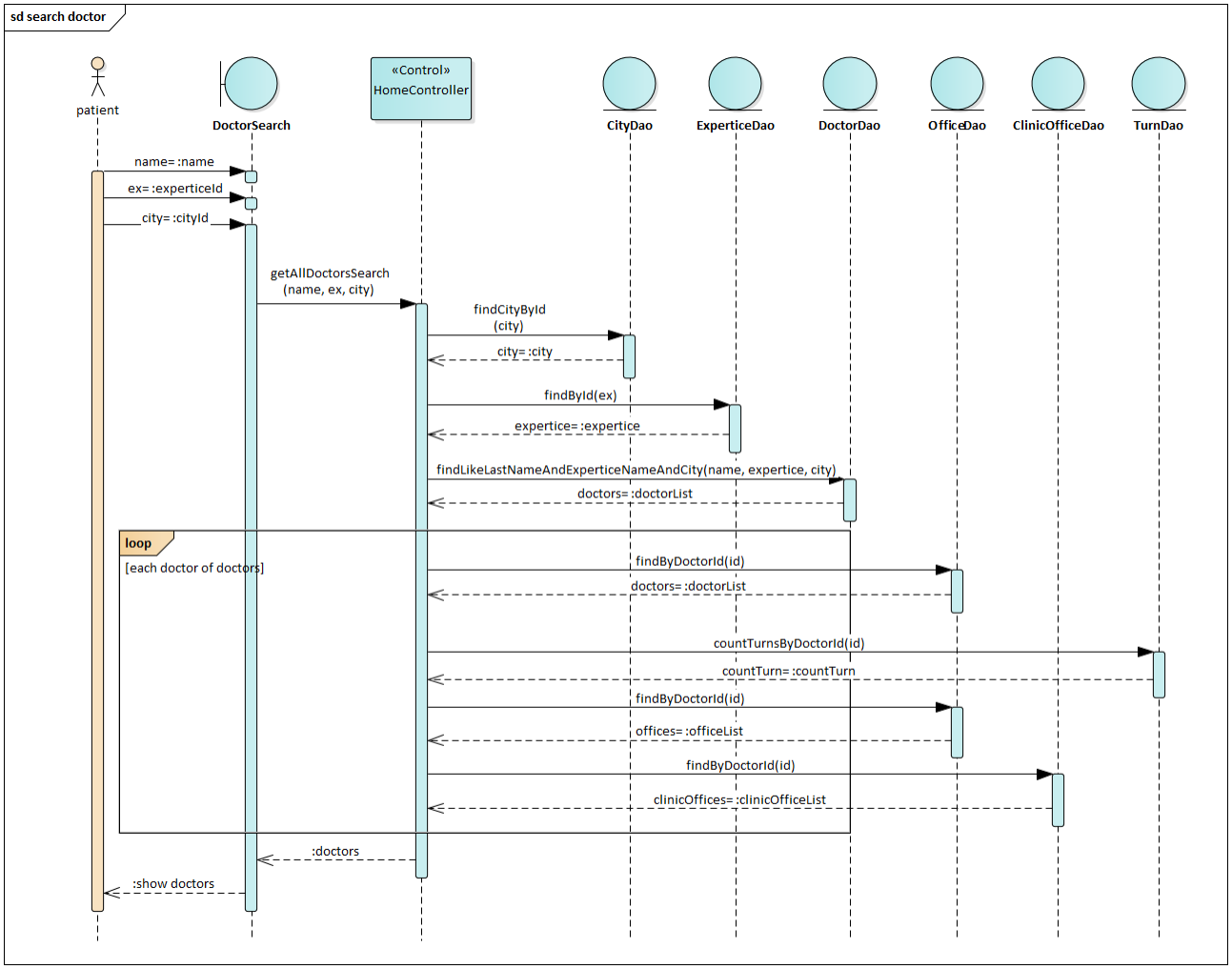
Use Case Number	۷
Use Case	لغو نوبت
Summary	کاربر میتواند نوبت هایی که هنوز ویزیت نشده اند یا لغو نشده اند را لغو کند
Actor	بیمار، پزشک، کلینیک
Trigger	دکمه لغو در نوبت های من
Primary Scenario	کاربر میتواند نوبت هایی که درحال انتظار هستند را لغو کند
Exceptional Scenario	بعد از جست و جو برای رزرو اگر کاربر وارد حساب خود نشده بود باید ابتدا به او پیام ابتدا باید وارد شوید را نمایش سپس به صفحه ورود انتقال پیدا کند
Pre-Conditions	کاربر به حساب کاربری خود وارد شده باشد



شکل ۳-۱۰ نمودار توالی لغو نوبت

جدول ۳-۸ مورد کاربرد جستجو پزشکان

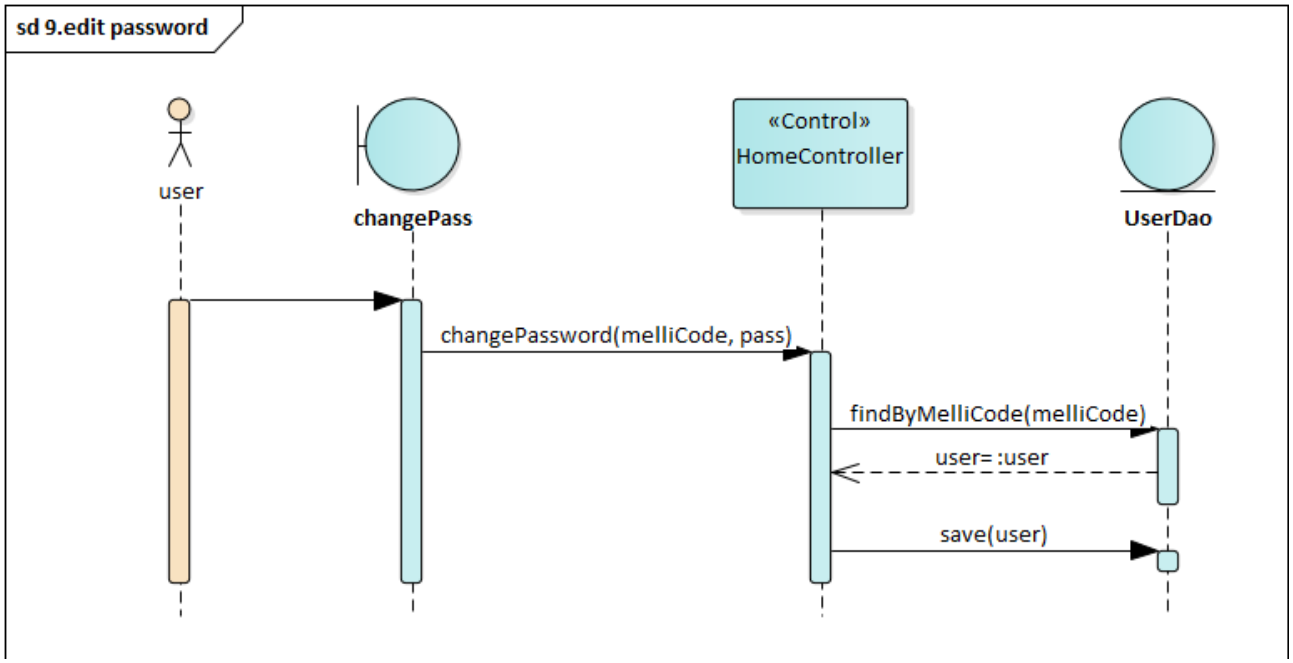
Use Case Number	۸
Use Case	جستجو پزشکان
Summary	بیماران میتوانند در سیستم جستجو کنند و پزشک مورد نظر خود را انتخاب کنند تا از او نوبت بگیرند
Actor	بیمار
Trigger	دکمه جستجو پزشک
Primary Scenario	بیمار میتواند بر اساس نام ، تخصص، و شهر پزشک مورد نظر خود در بین پزشکان جست و جو کند همچنین امکان فیلتر کردن براساس جنسیت و میزان تحصیلات پزشک نیز وجود دارد لیست به نمایش در امده به ترتیب تعداد نوبت های هر پزشک می باشد



شکل ۳-۱۱ نمودار توالی جستجو پزشکان

جدول ۳-۹ مورد کاربرد تغییر رمز عبور

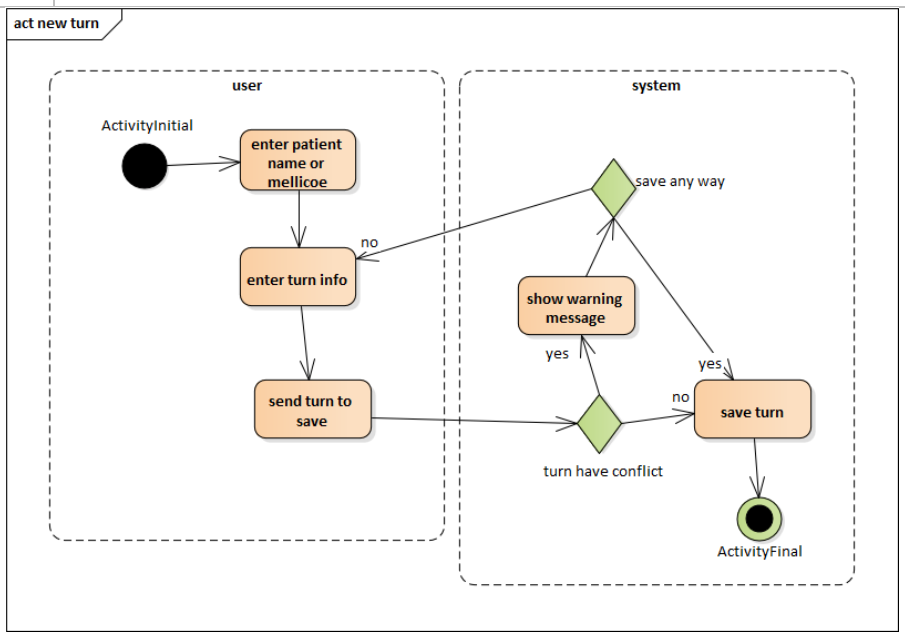
Use Case Number	۹
Use Case	تغییر رمز عبور
Summary	بیمار میتواند رمز عبور خود را تغییر دهد
Actor	بیمار، پزشک، کلینیک
Trigger	دکمه تغییر رمز عبور
Primary Scenario	کاربر با کلیک بر روی دکمه تغییر رمز عبور در فیلد مشخص شده رمز خود را وارد میکند و تایید میکند
Pre-Conditions	کاربر وارد سیستم شده باشد



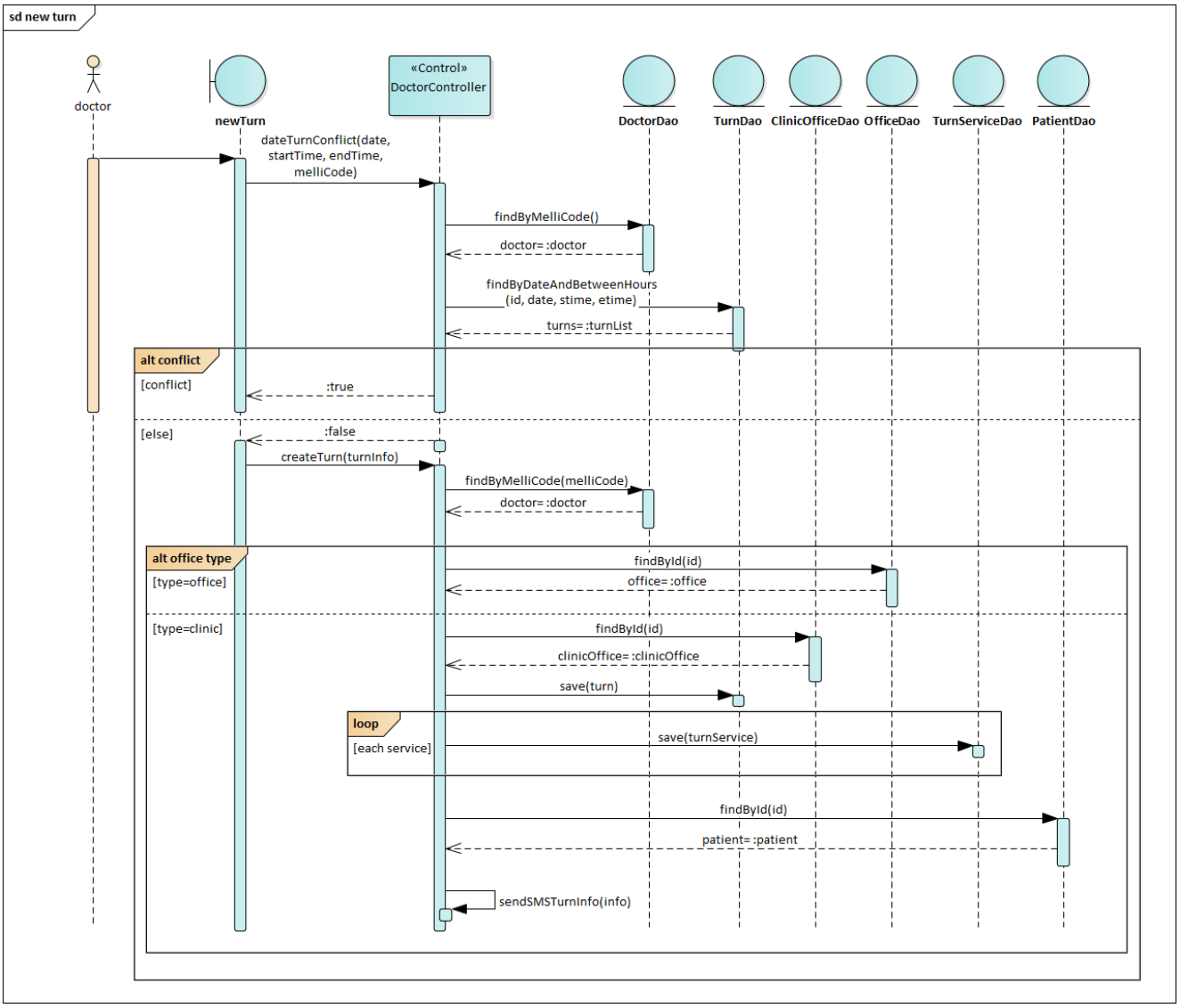
شکل ۳-۱۲ نمودار توالی تغییر رمز عبور

جدول ۳-۱۰ مورد کاربرد ثبت نوبت جدید

Use Case Number	۱۰
Use Case	ثبت نوبت جدید
Summary	پزشک میتواند برای بیماران نوبت جدید ثبت کند
Actor	پزشک
Trigger	کشیدن دکمه ثبت نوبت جدید در یک تاریخ
Primary Scenario	پزشک برای ثبت نوبت جدید دکمه ثبت نوبت جدید را کشیده و در تاریخ مورد نظر رها میکند سپس بر روی ان کلیک کرده فهرستی برای او باز میشود بر اساس نام خانوادگی یا کدملی، بیمار را جستجو می کند وفهرستی از بیماران با اطلاعات مشابه به او نمایش داده میشود یک بیمار را انتخاب میکند سپس ساعت شروع و پایان ویزیت را تعیین میکند و نوع درمان را براساس سرویس هایی که برای خود انتخاب کرده تعیین میکند و بیمه نیز از بین بیمه های مورد تایید خود یکی را انتخاب میکند و پزشک ممکن از در چندین کلینیک یا مطب کار کند برای همین کلینیک یا مطب مشخص میکند و میتواند هزینه و نیز توضیحاتی راجع به نوبت را بنویسد و نیز اگر لازم به اضافه کردن فایل هایی می باشد میتواند ان ها را به انتهای نوبت پیوست دهد
Exceptional Scenario	۱- نوبت دیگری در این تاریخ و ساعت برای این دکتر ثبت شده باشد در اینصورت به دکتر هشدار داده می شود و او میتواند نوبت را ثبت و از ثبت ان صرفنظر کند و تاریخ را تغییر دهد ۲- بیمار، نوع درمان، نوع بیمه ، انتخاب مطب/کلینیک نمی تواند خالی باشد و حتما باید انتخاب شوند
Pre-Conditions	۱- بیماری که پزشک میخواهد برای او نوبت بگیرید از قبل در سیستم ثبت نام کرده باشد ۲- پزشک وارد حساب کاربری خود شده باشد ۳- پزشک بیمه ها و سرویس های خود را قبلا ثبت کرده باشد ۴- پزشک ادرس حداقل یک مطب یا کیلینیک را در پروفایل خود ثبت کرده باشد



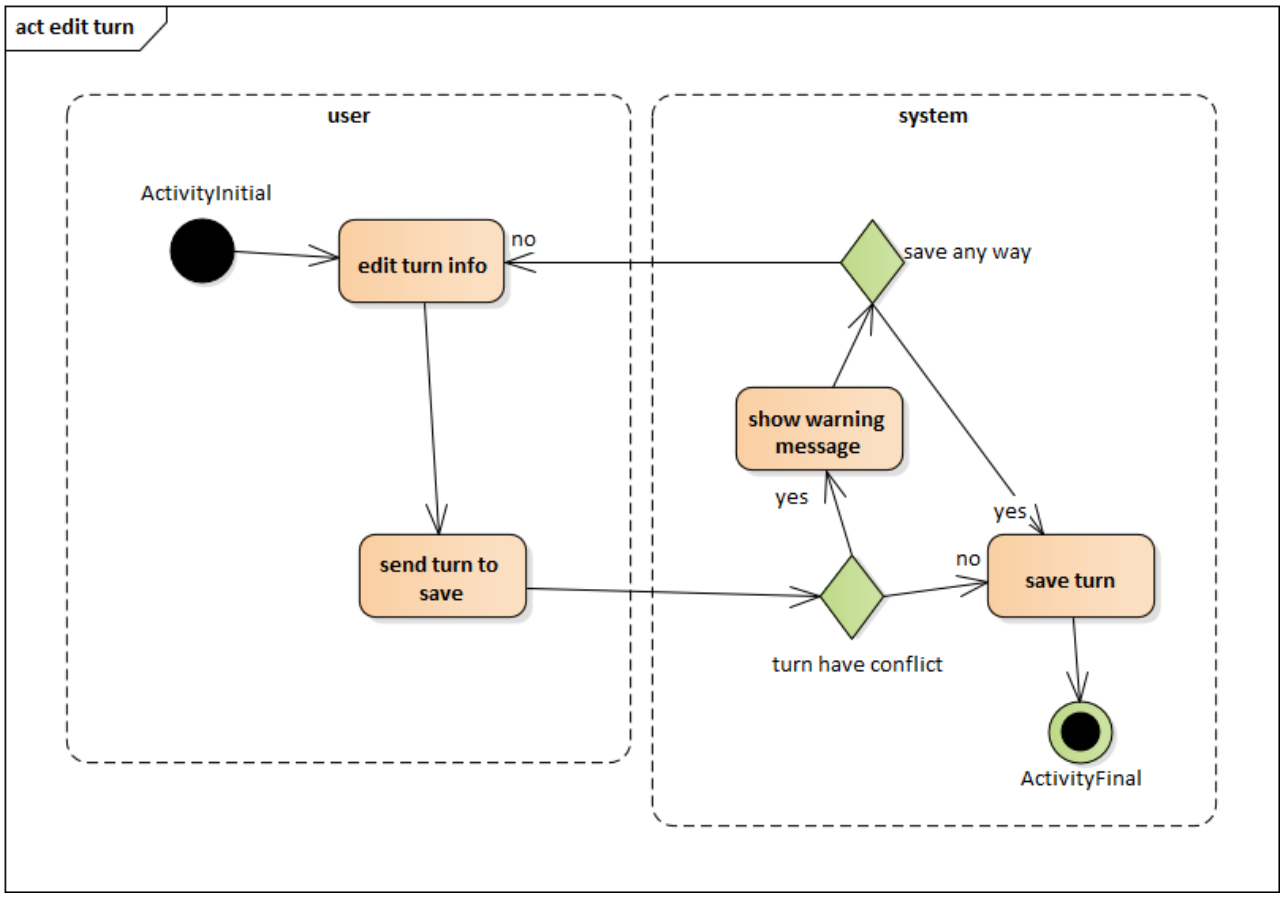
شکل ۳-۱۳ نمودار فعالیت ثبت نوبت جدید



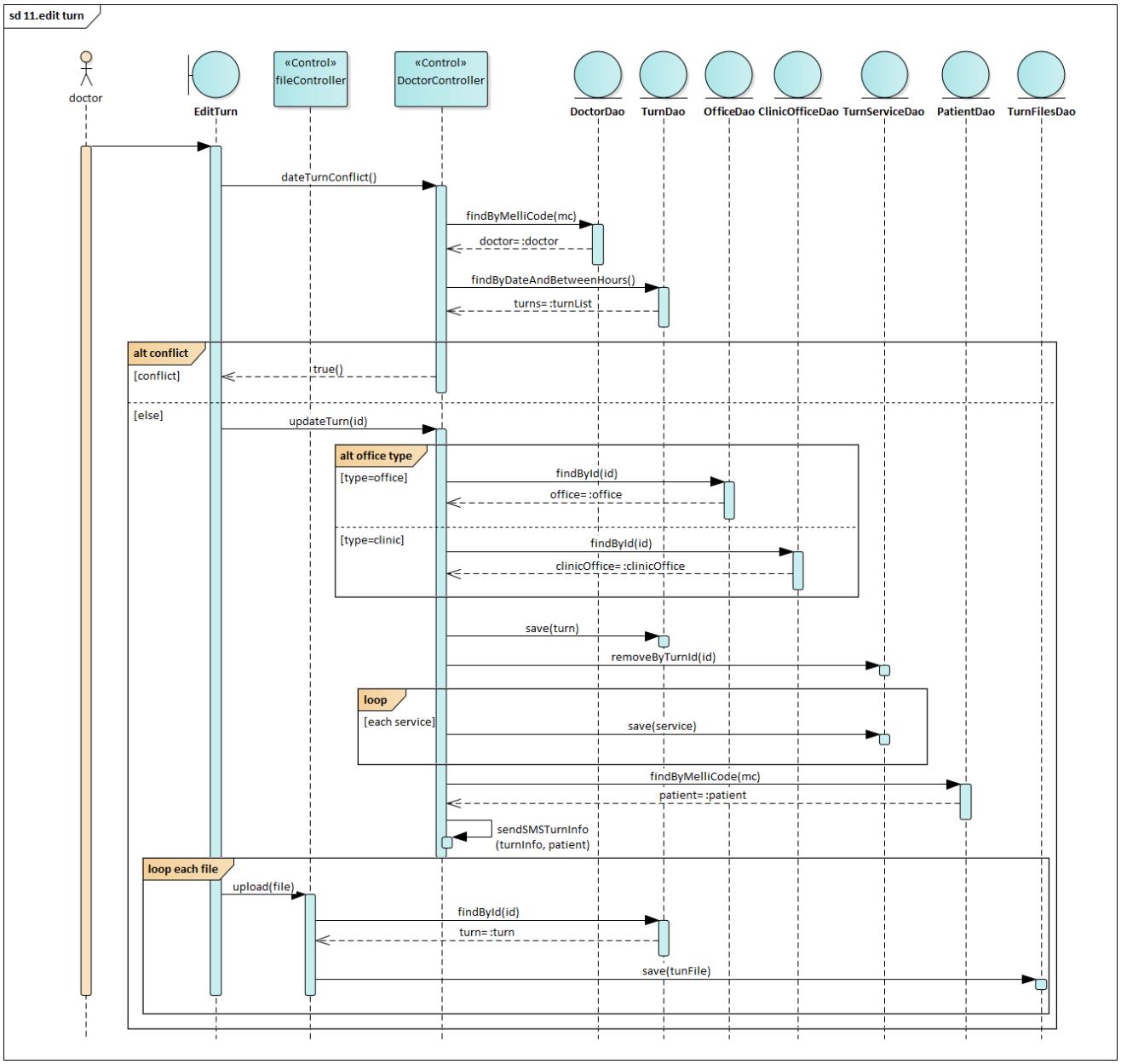
شکل ۳-۱۴ نمودار توالی ثبت نوبت جدید

جدول ۳-۱۱ مورد کاربرد تغییر اطلاعات نوبت

Use Case Number	۱۱
Use Case	تغییر اطلاعات نوبت
Summary	پزشک می تواند نوبت گرفته برای بیمار را تغییر دهد
Actor	پزشک
Trigger	کلیک بر روی هر نوبت در صفحه نوبت ها
Primary Scenario	با کلیک بر روی هر نوبت اطلاعات نوبت نمایش داده میشود پزشک میتواند تاریخ نوبت، ساعت شروع، ساعت پایان، نوع درمان، توضیحات، نوع بیمه، مطب/کلینیک، هزینه را تغییر دهد همچنین میتواند فایل های پیوست شده به نوبت را حذف یا اضافه کند هر نوبت ۳ وضعیت دارد درحال انتظار، ویزیت شده، لغو شده که پزشک میتواند نوبت را به هریک از این سه وضعیت ببرد
Exceptional Scenario	۱- اگر این نوبت با نوبت دیگری تداخل داشته باشد قبل از اعمال تغییرات هشداری داده میشود
Pre-Conditions	پزشک وارد حساب کاربری خود شده باشد



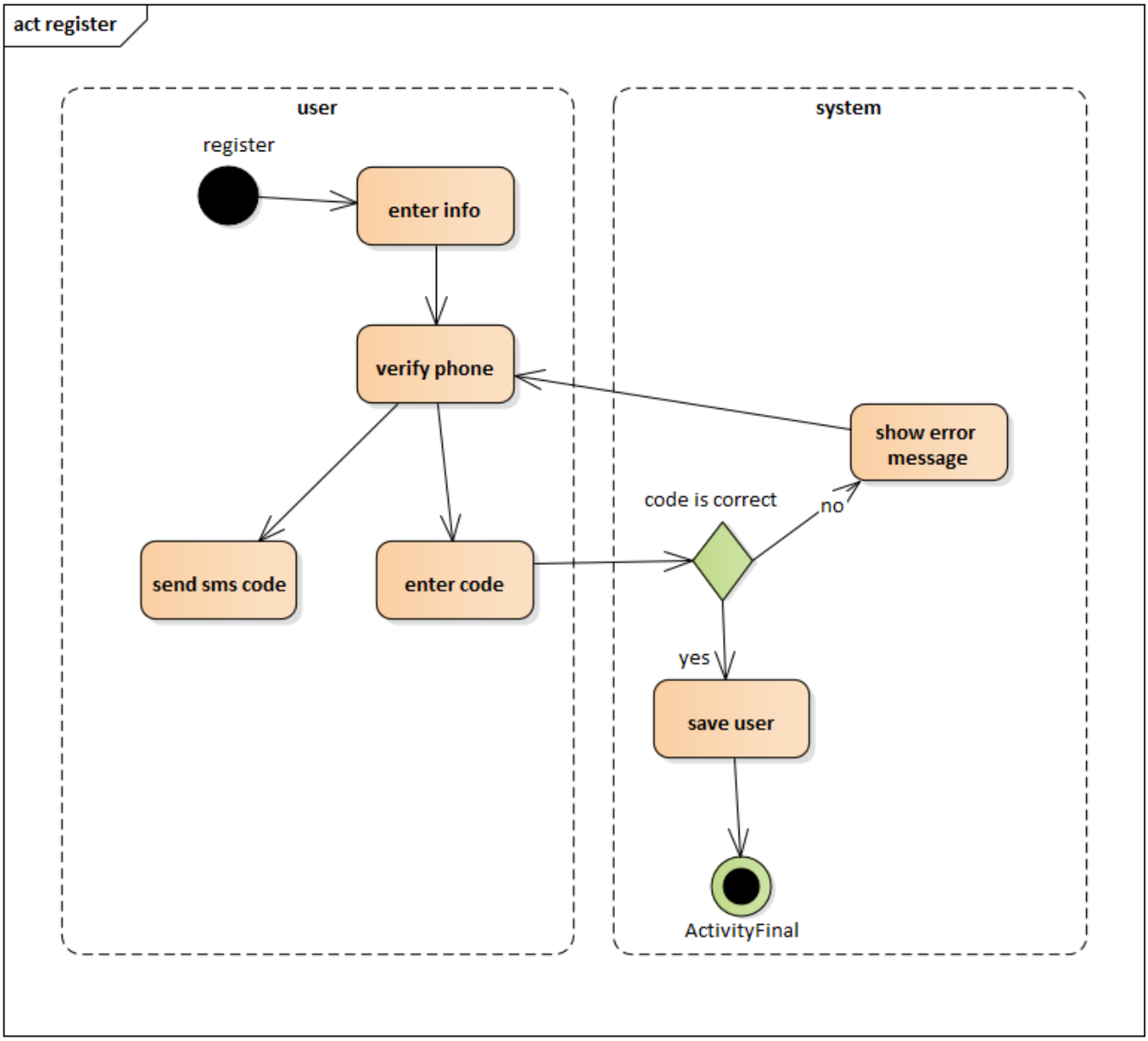
شکل ۳-۱۵ نمودار فعالیت تغییر اطلاعات نوبت



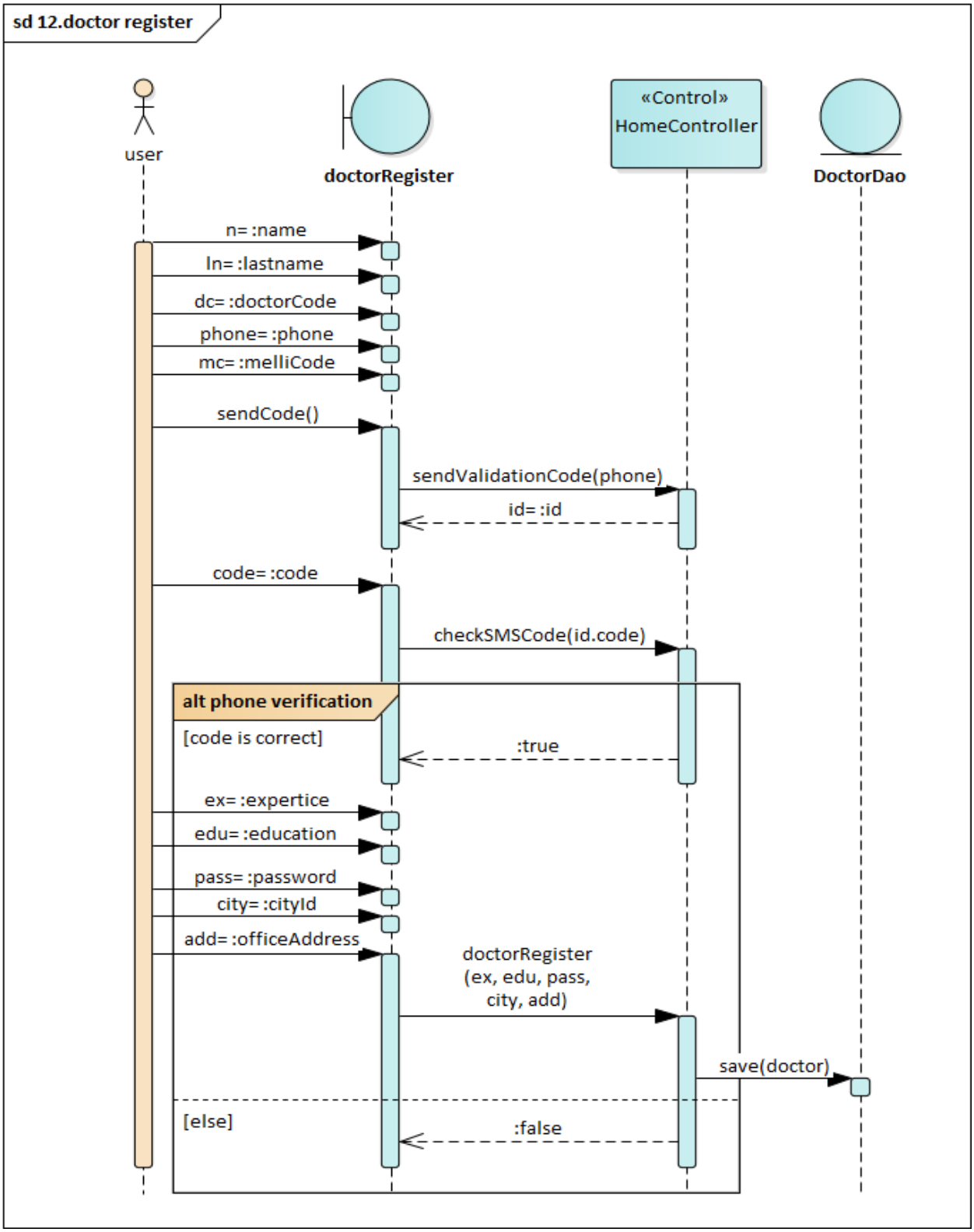
شکل ۳-۱۶ نمودار توالی تغییر اطلاعات نوبت

جدول ۳-۱۲ مورد کاربرد ثبت نام پزشک

Use Case Number	۱۲
Use Case	ثبت نام پزشک
Summary	پزشکان میتوانند در سیستم ثبت نام کنند
Actor	پزشک
Trigger	دکمه ثبت نام پزشک
Primary Scenario	پزشک باید اطلاعات زیر را وارد کند نام، نام خانوادگی، شماره نظام پزشکی، شماره ملی، شماره موبایل، جنسیت سپس باید به شماره موبایل او کد تایید فرستاده میشود در صورت تایید شماره پزشک باید اطلاعات زیر را وارد کند و سپس ثبت نام کند تخصص(تخصصش را باید از بین تخصص هایی که در پایگاه داده وجود دارد انتخاب کند)، تحصیلات، رمز عبور، ادرس مطب، شهر
Exceptional Scenario	۱- نام و نام خانوادگی باید فارسی باشد و حتما وارد شود در غیر اینصورت با پیام خطا مواجه خواهد شد ۲- شماره ملی باید ۱۰ رقمی باشد و نباید تمام ارقام ان یکسان باشد و نیز نمی تواند خالی باشد ۳- شماره موبایل باید ۱۱ رقمی باشد و نیز نمی تواند خالی باشد ۴- تخصص و تحصیلات و شهر نمی تواند خالی باشد ۵- شماره نظام پزشکی باید ۵ رقمی و نمی تواند خالی باشد
Pre-Conditions	۱- پزشک شماره موبایل معتبر داشته باشد ۲- پزشک کدملی و شماره نظام پزشکی داشته باشد
Post-Conditions	۱- شماره ملی تکراری نباشد ۲- موبایل تکراری نباشد ۳- شماره نظام پزشکی تکراری نباشد



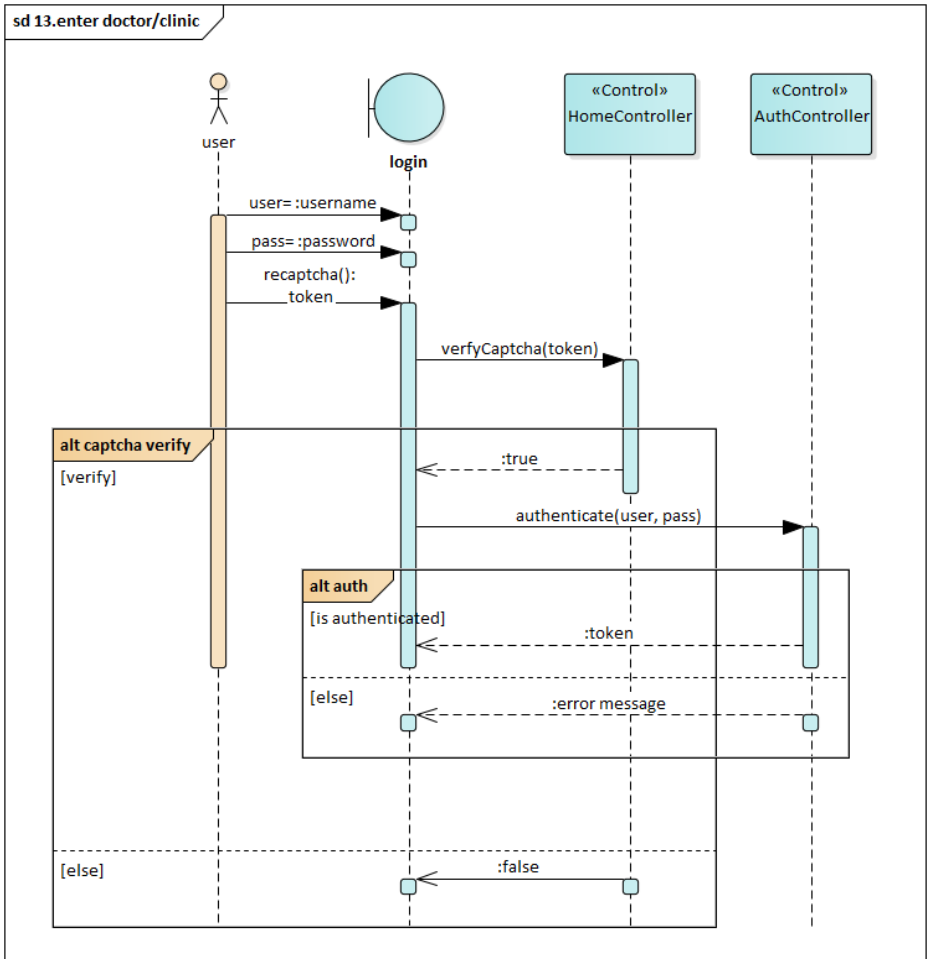
شکل ۳-۱۷ نمودار فعالیت ثبت نام پزشک



شکل ۳-۱۸ نمودار توالی ثبت نام پزشک

جدول ۳-۱۳ مورد کاربرد ورود پزشک و کلینیک

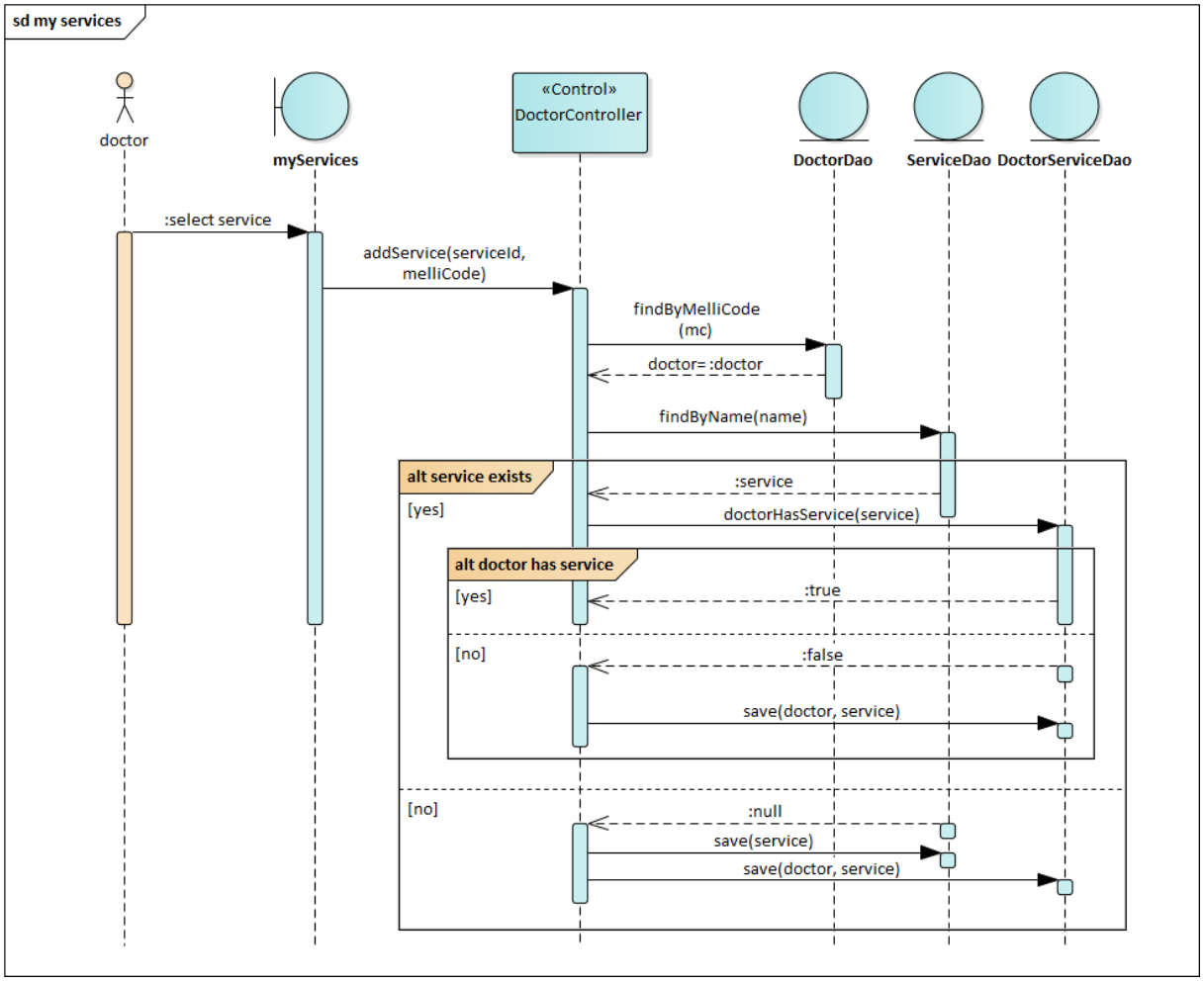
Use Case Number	۱۳
Use Case	ورود پزشک و کلینیک
Summary	کاربر برای استفاده از سیستم باید به ان وارد شوند
Actor	پزشک، کلینیک
Trigger	دکمه ورود پزشک
Primary Scenario	کاربر شماره ملی و رمز عبور خود را وارد میکند سپس ریکپچا را تایید میکند و بر روی ورود کلیک میکند درصورت درست بودن اطلاعات توکن به کاربر داده میشود که با ان میتواند به قسمت های تعیین شده برای او دسترسی دشته باشد
Alternative Scenario	اگر کاربر قبلا وارد شده باشد و توکن او هنوز معتبر باشد نیازی به این قسمت ندارد
Exceptional Scenario	۱- در صورت نادرست بودن رمز عبور یا موجود نبودن کدملی در سیستم پیام خطا به کاربر نمایش داده میشود ۲- درصورت تایید نکردن ریکپچا کاربر قدر به ورود نمی باشد
Pre-Conditions	کاربر حساب کاربری داشته باشد



شکل ۳-۱۹ نمودار توالی ورود پزشک و کلینیک

جدول ۳-۱۴ مورد کاربرد سرویس های من

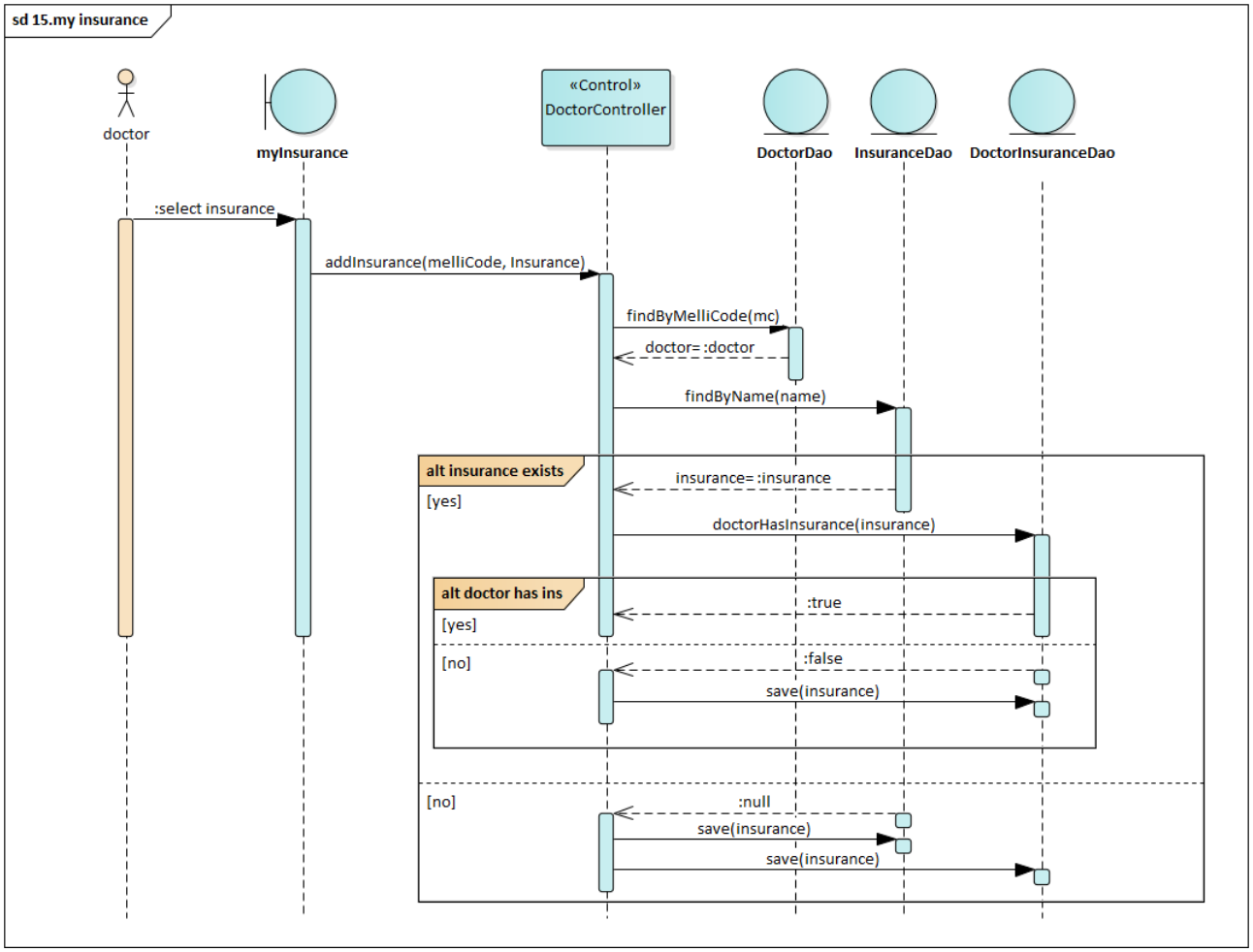
Use Case Number	۱۴
Use Case	سرویس های من
Summary	هر پزشک بنا به تخصص خود چندین سرویس ارائه میدهد برای مثال دندانپزشک سرویس هایی چون عصب کشی، پرکردن و... را دارد
Actor	پزشک
Trigger	دکمه سرویس های ارائه شده در پنل پزشک
Primary Scenario	پزشک از قسمت مشخص شده میتواند سرویس هایی که ارائه میدهد را جستجو کند و یک یا چند تا از ان ها را برای خود انتخاب کند یا در صورت موجود نبودن سرویس مورد نظر پزشک در سیستم ان ها را به سیستم اضافه کند و برای خود انتخاب کند
Exceptional Scenario	۱- سرویس هایی که در سیستم ثبت شده اند نمی توانند دوباره ثبت شوند و فقط سرویس هایی که در سیستم وجود ندارند میتوانند در سیستم ثبت شوند
Pre-Conditions	پزشک به حساب کاربری خود وارد شده باشد
Post-Conditions	اگر هیچ سرویسی انتخاب نشود پزشک نمیتواند هیچ نوبتی ثبت کند



شکل ۳-۲۰ نمودار توالی سرویس های من

جدول ۳-۱۵ مورد کاربرد بیمه های من

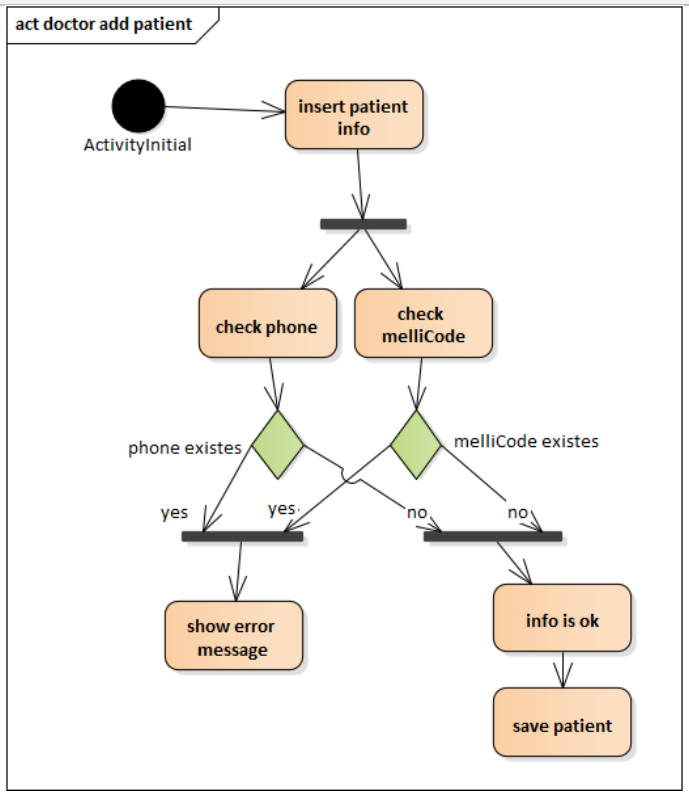
Use Case Number	۱۵
Use Case	بیمه های من
Summary	هر پزشک باید تعیین کند که از چه بیمه هایی استفاده میکند
Actor	پزشک
Trigger	دکمه سروریس های ارائه شده
Primary Scenario	در کادر مشخص شده پزشک باید لیست بیمه هایی که با انها طرف قرارداد می باشد را وارد کند
Exceptional Scenario	۱- بیمه هایی که ثبت شده اند نمی توانند دوباره ثبت شوند
Pre-Conditions	پزشک باید به حساب کاربری خود وارد شده باشد
Post-Conditions	اگر هیچ بیمه ای انتخاب نشود پزشک نمی تواند نوبتی ثبت کند



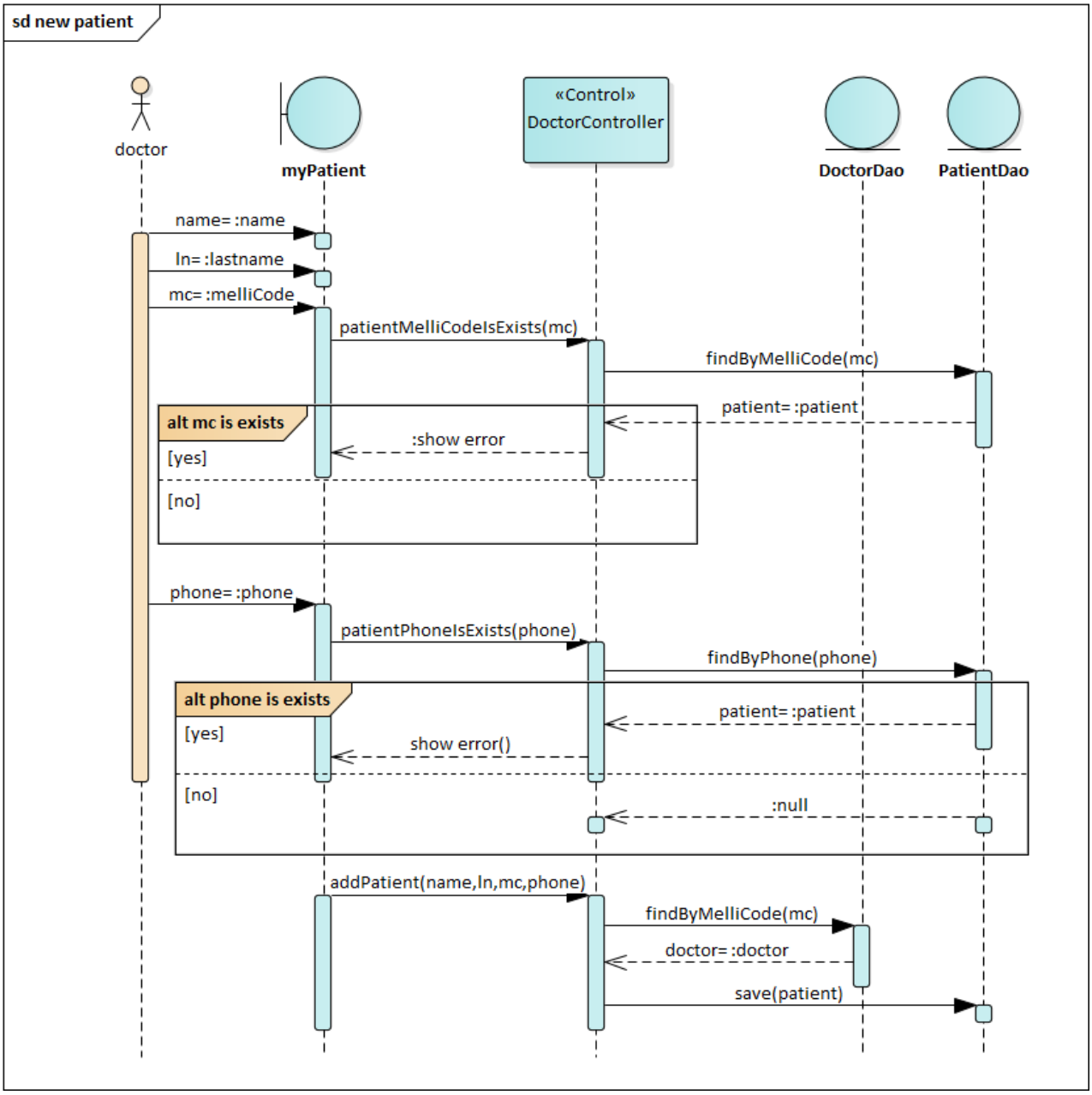
شکل ۳-۲۱ نمودار توالی بیمه های من

جدول ۳-۱۶ مورد کاربرد اضافه کردن بیمار

Use Case Number	۱۶
Use Case	اضافه کردن بیمار
Summary	پزشک میتواند درصورتی که بیمار در سیستم حساب کاربری نداشته باشد برای او یک حساب ایجاد کند تا بتواند برایش نوبت بگیرد
Actor	پزشک
Trigger	دکمه ثبت بیمار جدید در صفحه بیماران من
Primary Scenario	در کادر نمایش داده شده پزشک باید نام ، نام خانوادگی، کدملی، شماره تلفن، جنسیت بیمار را وارد کند سپس بیمار را ثبت کند در این حالت حساب بیمار رمز عبور ندارد و هیچ کس قادر به دسترسی به حساب کاربری بیمار نیست بیمار برای دسترسی به حساب کاربری خود باید در قسمت فراموشی رمز عبور برای خود رمزی انتخاب کند تا قادر به دسترسی به حساب خود باشد
Alternative Scenario	بیمار خود حساب کاربری ایجاد کند
Exceptional Scenario	۱-اگر کد ملی تکراری باشد پیام خطا نمایش داده میشود ۲- اگر شماره تلفن تکراری باشد پیام خطا نمایش داده میشود
Pre-Conditions	پزشک باید به حساب کاربری خود وارد شده باشد
Post-Conditions	۱- اگر کدملی بیمار تکراری باشد ثبت نمی شود ۲- اگر شماره تلفن بیمار تکراری باشد ثبت نمی شود



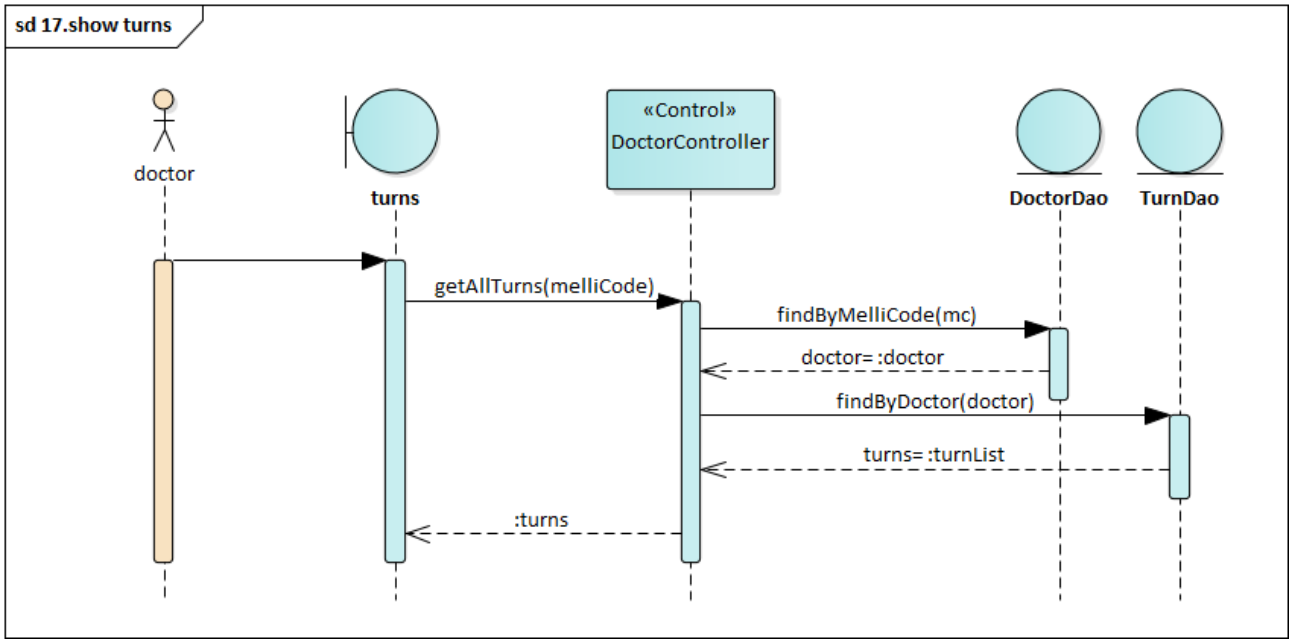
شکل ۳-۲۲ نمودار فعالیت اضافه کردن بیمار



شکل ۳-۲۳ نمودار توالی اضافه کردن بیمار

جدول ۳-۱۷ مورد کاربرد نمایش نوبت ها

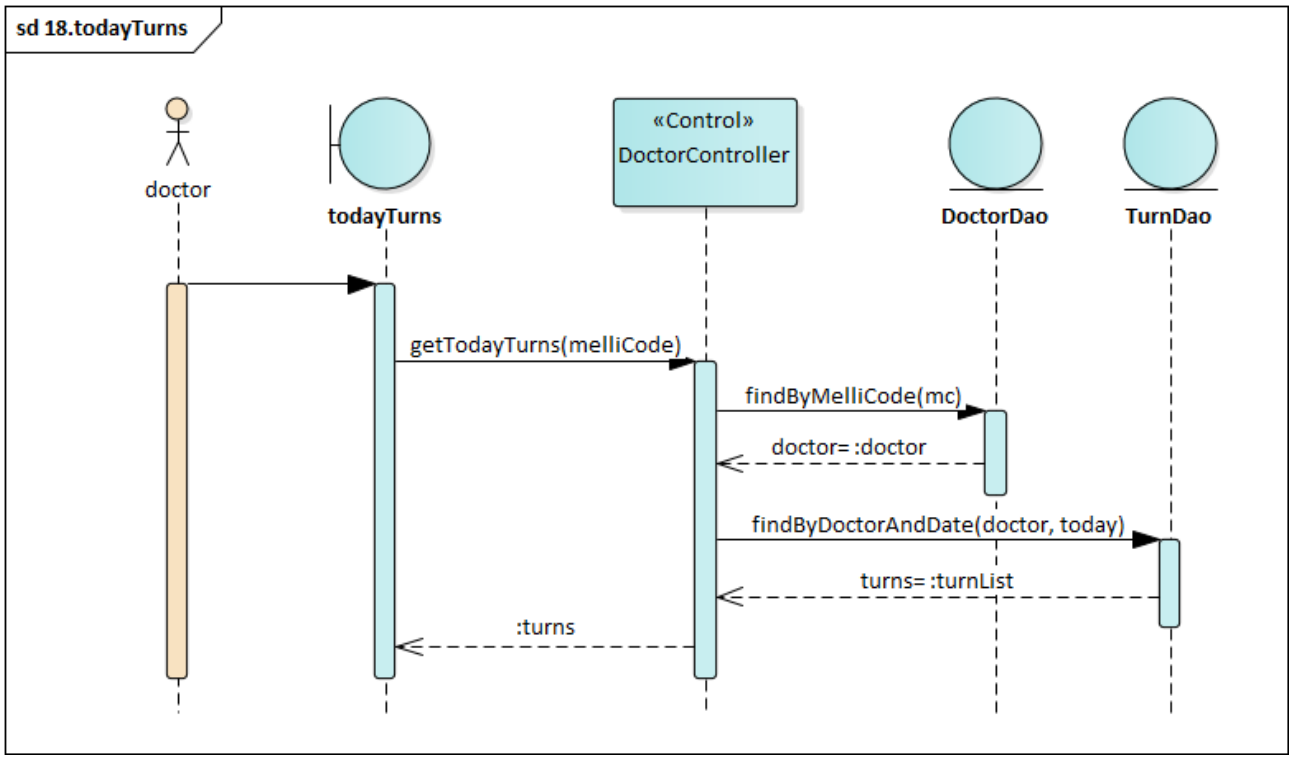
Use Case Number	۱۷
Use Case	نمایش نوبت ها
Summary	هر پزشک میتواند نوبت هایی که به بیماران مختلف داده است را مشاهده کند
Actor	پزشک
Trigger	دکمه نوبت ها
Primary Scenario	تمام نوبت ها بصورت ماهانه و بصورت تقویمی قابل مشاهده است و پزشک میتواند با کلیک بر روی هر نوبت در هر تاریخ جزئیات ان را مشاهده کند هر نوبت میتواند۳ رنگ داشته باشد رنگ ابی: نوبت های درحال انتظار، رنگ قرمز: نوبت های لغو شده، رنگ سبز: نوبت های ویزیت شده می باشد
Pre-Conditions	کاربر باید به حساب کاربری خود وارد شده باشد



شکل ۳-۲۴ نمودار توالی نمایش نوبت ها

جدول ۳-۱۸ مورد کاربرد نمایش نوبت های امروز

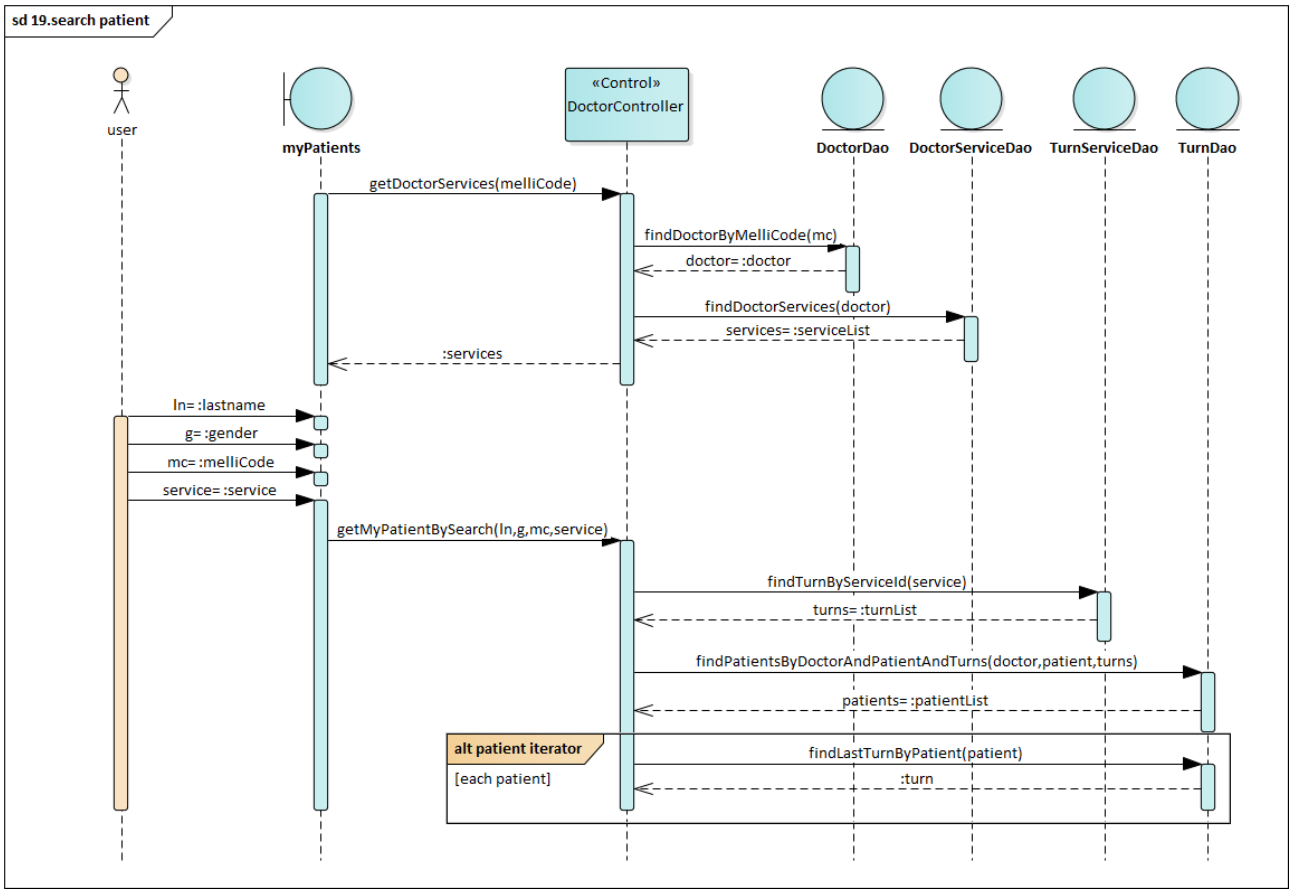
Use Case Number	۱۸
Use Case	نمایش نوبت های امروز
Summary	برای سادگی کار، پزشکان میتوانند در این قسمت نوبت های روز خود را مشاهده کنند
Actor	پزشک
Trigger	دکمه نوبت های امروز
Primary Scenario	در این قسمت به ترتیب ساعت نوبت ها نمایش داده میشود اطلاعات نمایش داده شده از هر نوبت شامل: نام و نام خانوادگی بیمار، تاریخ، ساعت، نوع درمان، توضیحات، بیمه، هزینه و فایل های پیوست است که هر یک قابل تغییر می باشند برای هر نوبت پزشک میتواند نوبت را لغو یا ویزیت کند که در هر دو صورت از لیست نوبت های امروز او خارج میشود
Pre-Conditions	پزشک به حساب کاربری خود وارد شده باشد نوبتی برای ان روز وجود داشته باشد
Post-Conditions	



شکل ۳-۲۵ نمودار توالی نمایش نوبت های روز

جدول ۳-۱۹ مورد کاربرد جستجو بیماران توسط پزشک

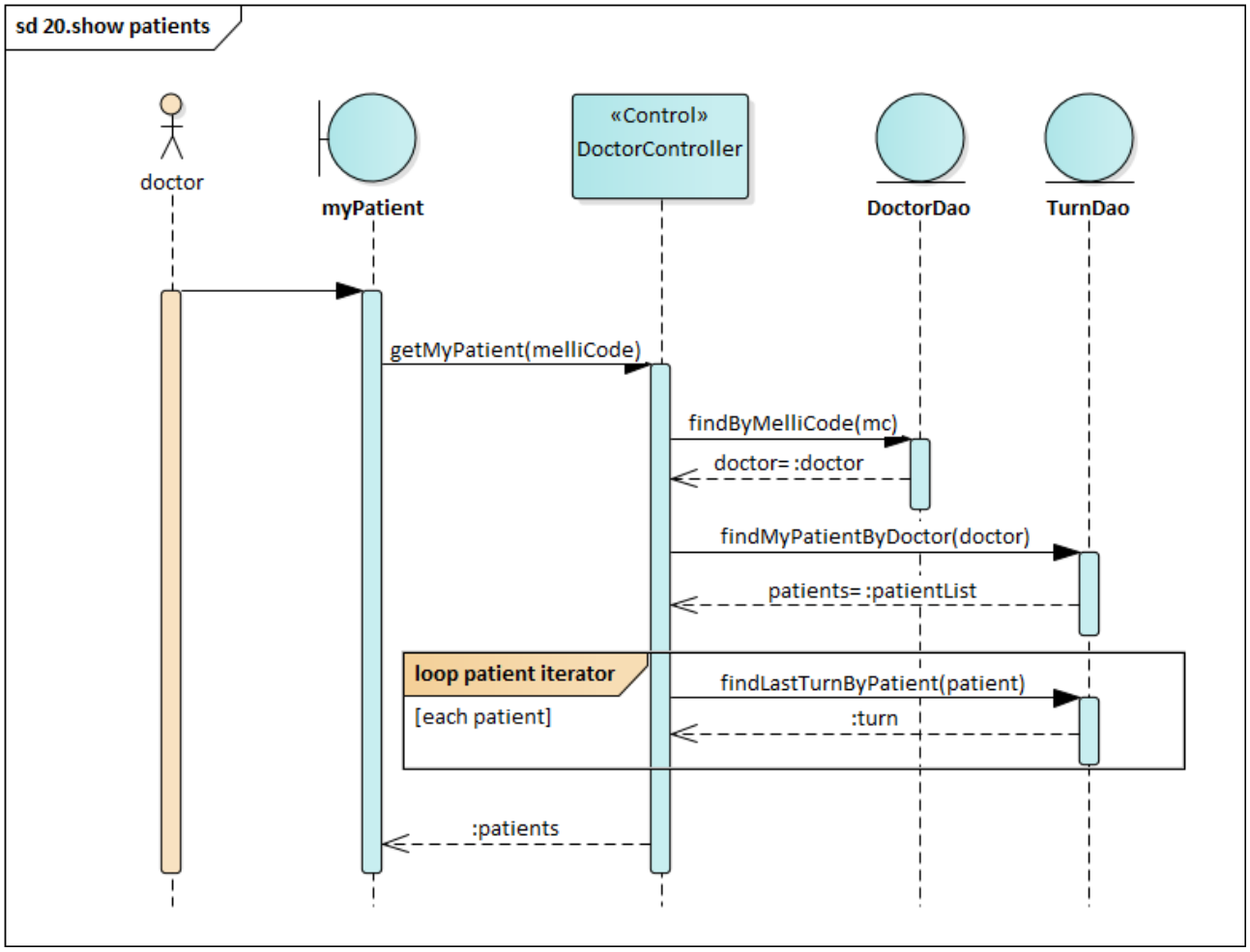
Use Case Number	۱۹
Use Case	جستجو بیماران توسط پزشک
Summary	پزشک میتواند از بین بیمارانی که به انها مراجعه کرده اند جستجو کنند و اطلاعات کاربری یا سابقه انها را مشاهده کنند
Actor	پزشک
Trigger	دکمه بیماران من
Primary Scenario	پزشک میتواند در بین بیمارانی که به او مراجعه کرده اند از طریق کدملی، جنسیت، نام خانوادگی، نوع درمان جستجو کند
Pre-Conditions	۱- پزشک حداقل یک نوبت داشته باشد ۲- پزشک به حساب کاربری خود وارد شده باشد



شکل ۳-۲۵،۱ نمودار توالی جستجو بیماران توسط پزشک

جدول ۳-۲۰ مورد کاربرد نمایش بیماران

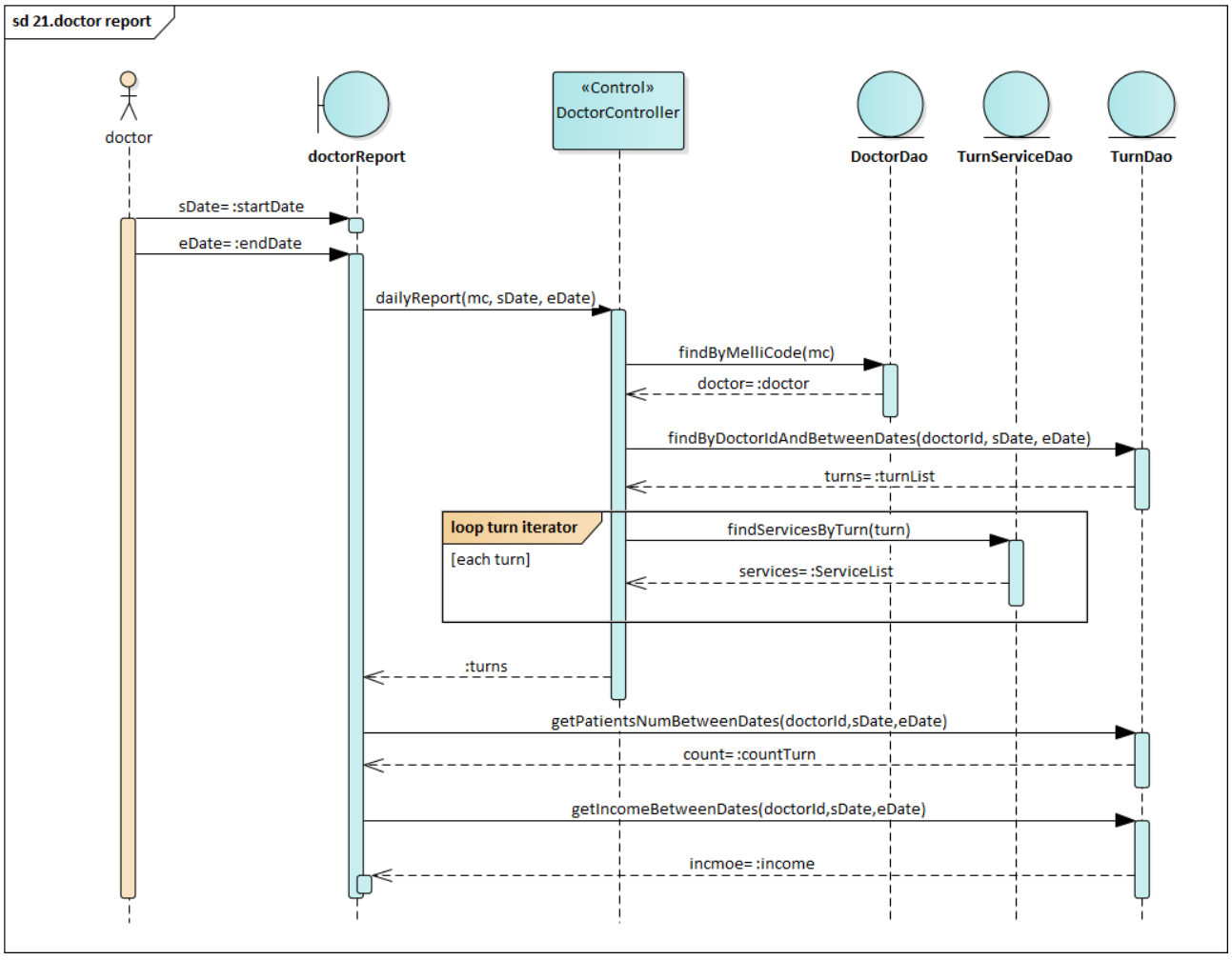
Use Case Number	۲۰
Use Case	نمایش بیماران
Summary	پزشکان و کلینک ها اطلاعات بیماران خود را مشاهده کنند
Actor	پزشک ، کلینیک
Trigger	دکمه بیماران من
Primary Scenario	پزشک و کلینیک میتواند اطلاعات بیماران خود را که شامل : نام و نام خانوادگی، کدملی، شماره موبایل، تاریخ و ساعت اخرین نوبت می باشد را مشاهده کنند
Pre-Conditions	۱- حدقل یک نوبت ثبت شده باشد ۲- کاربر به حساب کاربری خود وارد شده باشد



شکل ۳-۲۶ نمودار توالی نمایش بیماران

جدول ۳-۲۱ مورد کاربرد گزارش گیری پزشک

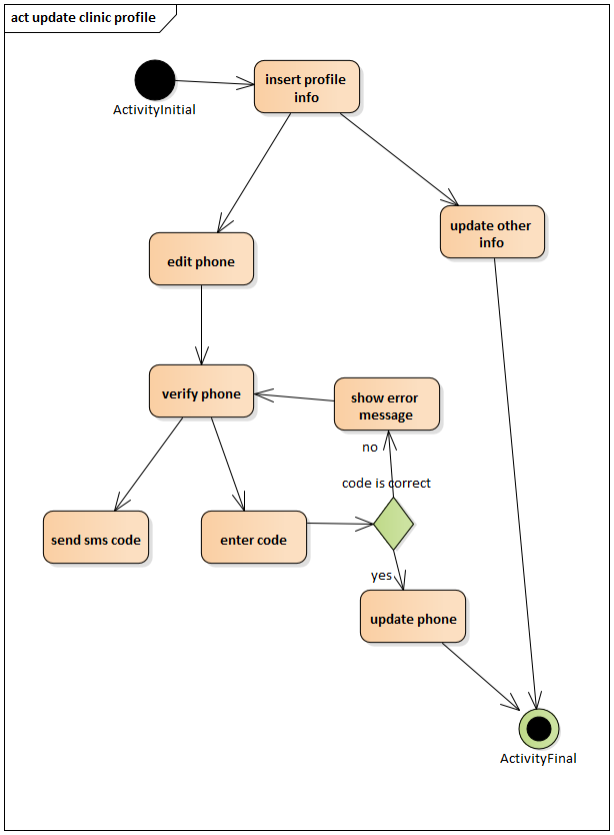
Use Case Number	۲۱
Use Case	گزارش پزشک از نوبت ها
Summary	پزشکان میتوانند در بازه مشخص از نوبت های خود گزارش تهیه کنند
Actor	پزشک
Trigger	دکمه گزارش روزانه
Primary Scenario	پزشک دو تاریخ را مشخص میکند بصورت گرافیکی نموداری تعداد نوبت های پزشک را براساس تاریخ نشان میدهد همچنین در جدولی جزئیات هر نوبت شامل تاریخ، نام بیمار، جنیست، نوع درمان، وضعیت نوبت، مبلغ نمایش داده میشود
Pre-Conditions	۱- حدقل یک نوبت در بازه مشخص شده ثبت شده باشد ۲- پزشک به حساب کاربری خود وارد شده باشد



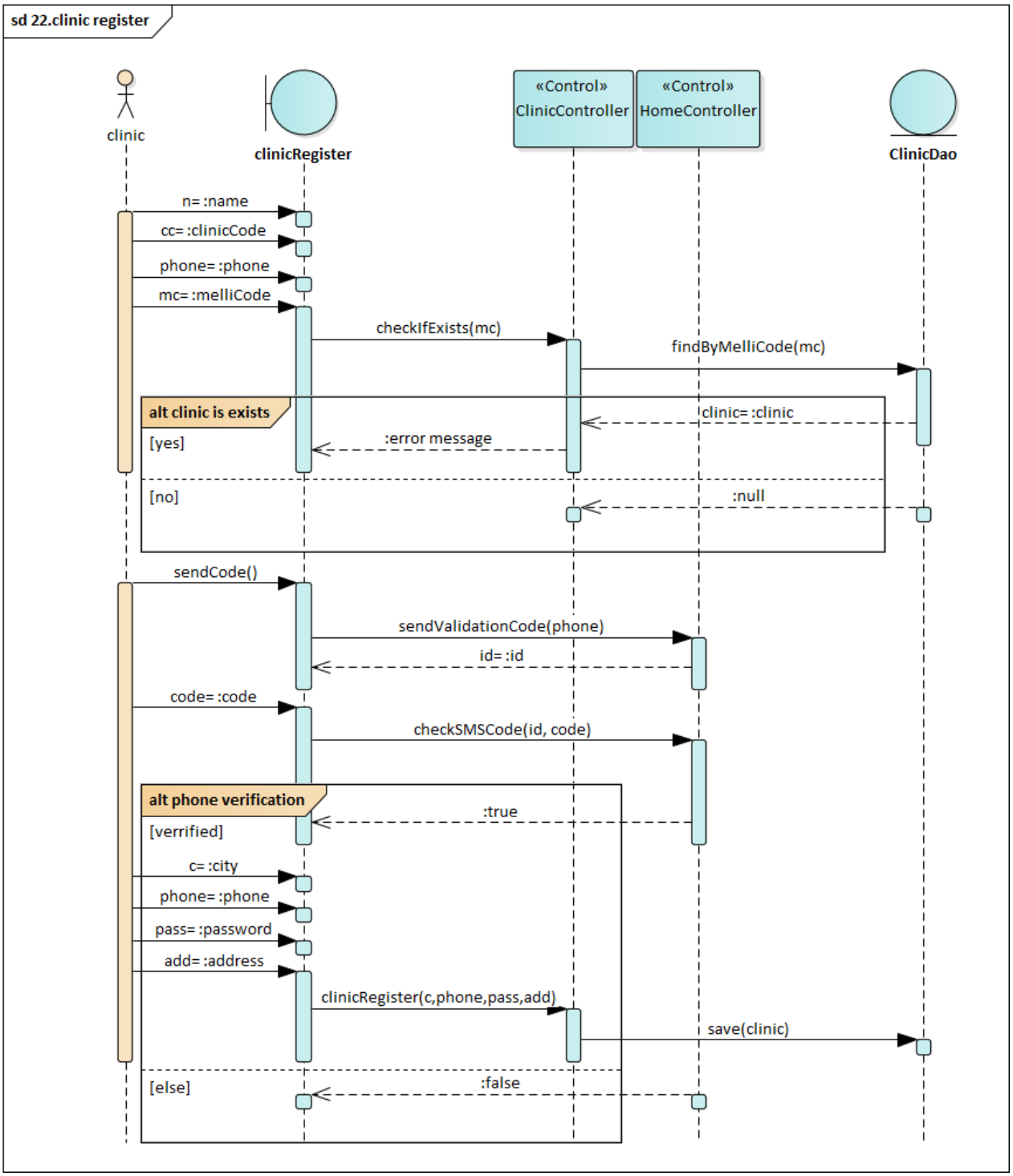
شکل ۳-۲۷ نمودار توالی گزارش گیری پزشک

جدول ۳-۲۲ مورد کاربرد ثبت نام کلینیک

Use Case Number	۲۲
Use Case	ثبت نام کلینیک
Summary	هر کلینیک برای استفاده از سیستم باید در ان ثبت نام کند
Actor	کلینیک
Trigger	دکمه ثبت نام کلینیک
Primary Scenario	هر کلینیک برای ثبت نام باید اطلاعات زیر را وارد کند. نام کلینیک، شماره ثبت، شماره موبایل، شماره ملی را وارد کند سپس باید شماره موبایل خود را با کد تاییدی که برایش پیامک می شود تایید کند بعد باید شهر، تلفن کلینیک، رمز عبور، و ادرس کلینیک را وارد کند را وارد کند تا کلینیک در سیستم ثبت شود
Exceptional Scenario	۱- اگر شماره ملی موجود باشد پیام خطا نمایش داده میشود ۲- نام کلینیک باید فارسی انتخاب شود
Pre-Conditions	کلینیک باید شماره ثبت، شماره تلفن شماره موبایل برای دسترسی به حساب خود داشته باشد
Post-Conditions	۱- اگر از قبل ثبت نام کرده باشد دوباره ثبت نام نخواهد شد ۲- اگر شماره ملی ، موبایل، شماره ثبت تکراری باشد ثبت نخواهد شد



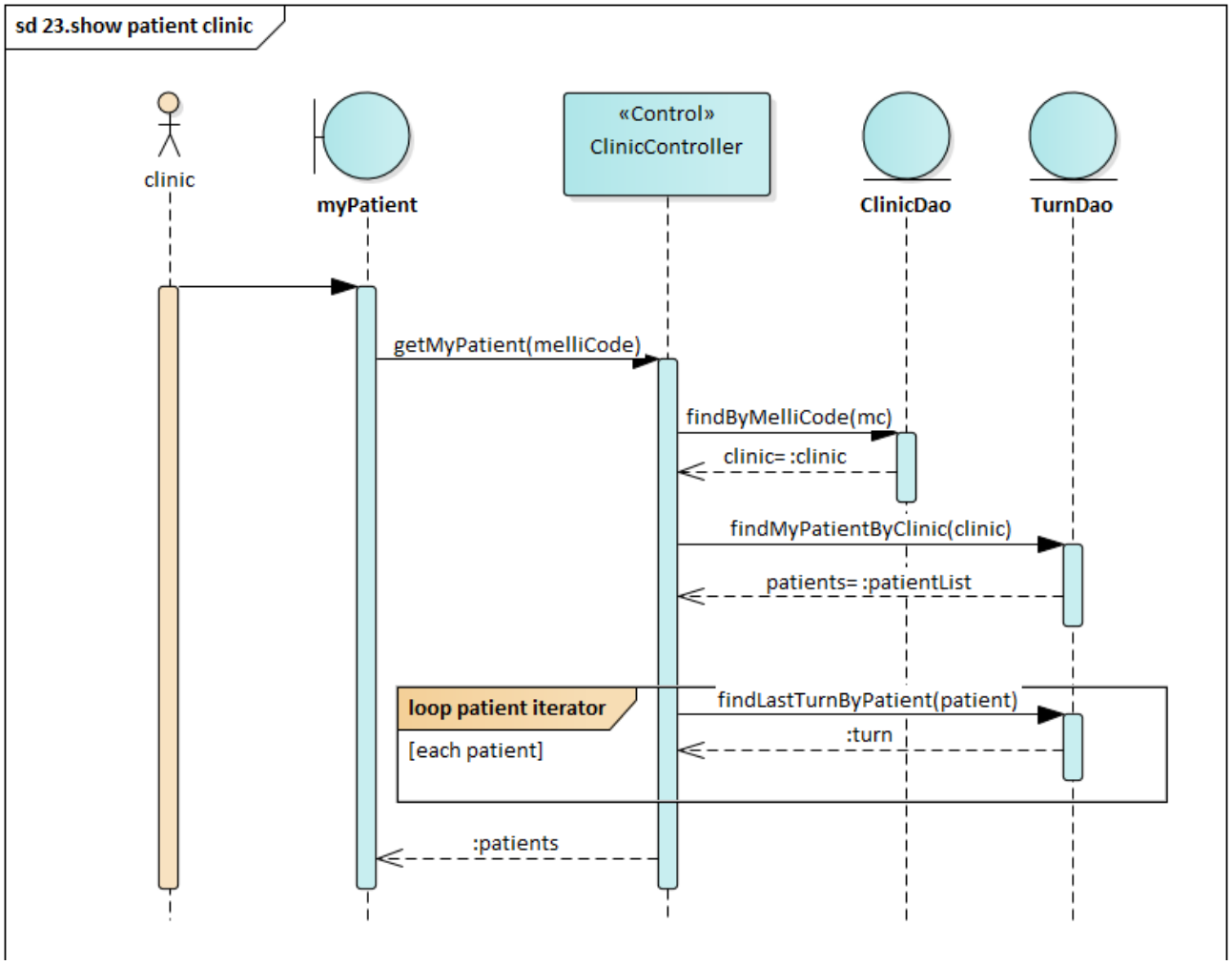
شکل ۳-۲۸ نمودار فعالیت ثبت نام کلینیک



شکل ۳-۲۹ نمودار توالی ثبت نام کلینیک

جدول ۳-۲۳ مورد کاربرد نمایش بیماران کلینیک

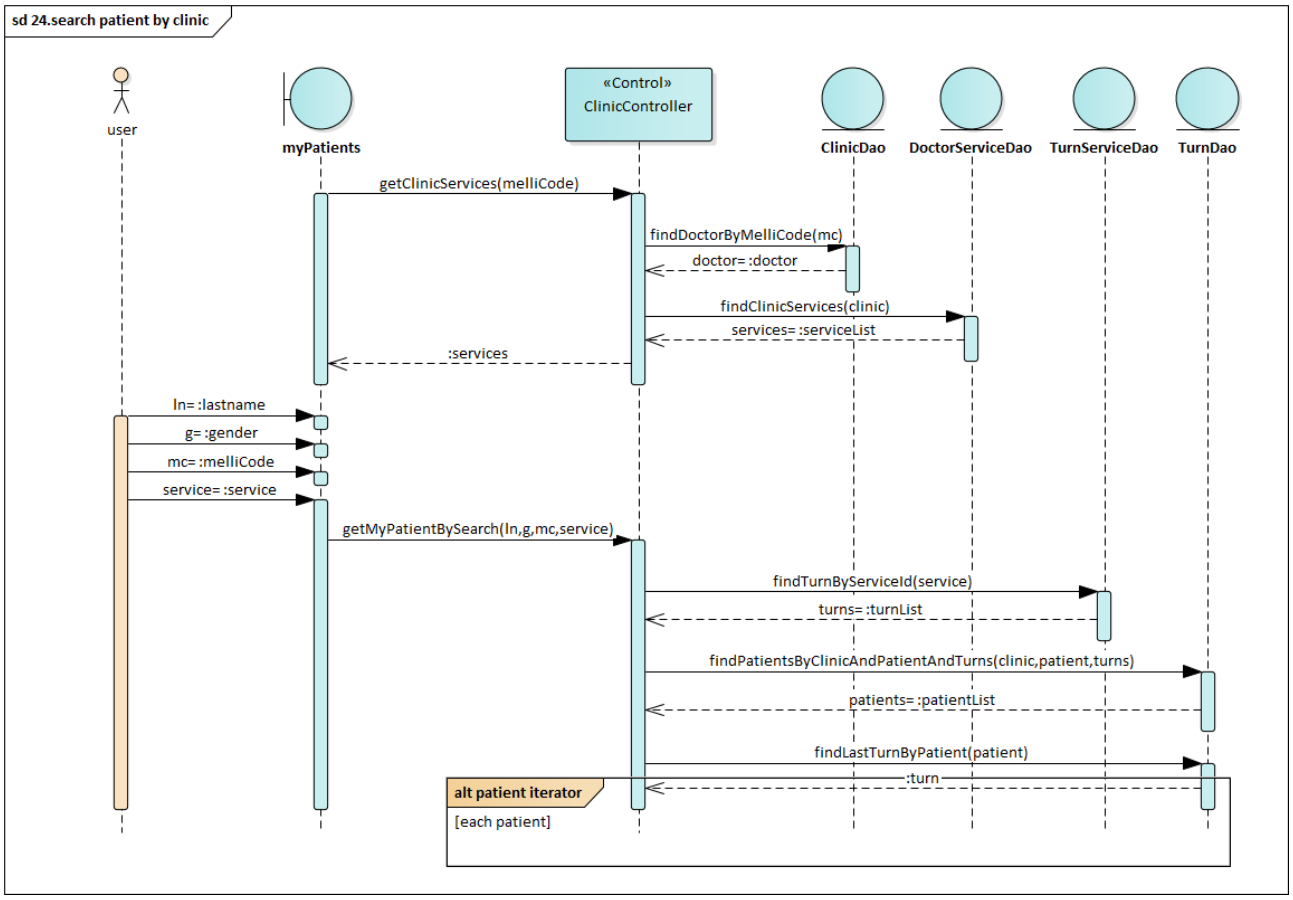
Use Case Number	۲۳
Use Case	نمایش بیماران کلینیک
Summary	کلینیک میتوانند بیمارانی که به کلینیک او مراجعه کرده اند مشاهده کند
Actor	کلینیک
Trigger	دکمه بیماران من
Primary Scenario	هر بیماری که حداقل یک بار در کلینیک از هر پزشک عضو کلینیک نوبت گرفته باشد در لیست بیماران کلینیک می باشد و کلینیک میتواند مشخصات او را مشاهده کند
Pre-Conditions	۱-کلینیک به حساب خود وارد شده باشد ۲ حداقل یک نوبت در کلینیک ثبت شده باشد



شکل ۳-۳۰ نمودار توالی نمایش بیماران کلینیک

جدول ۳-۲۴ مورد کاربرد جستجو بیماران کلینیک

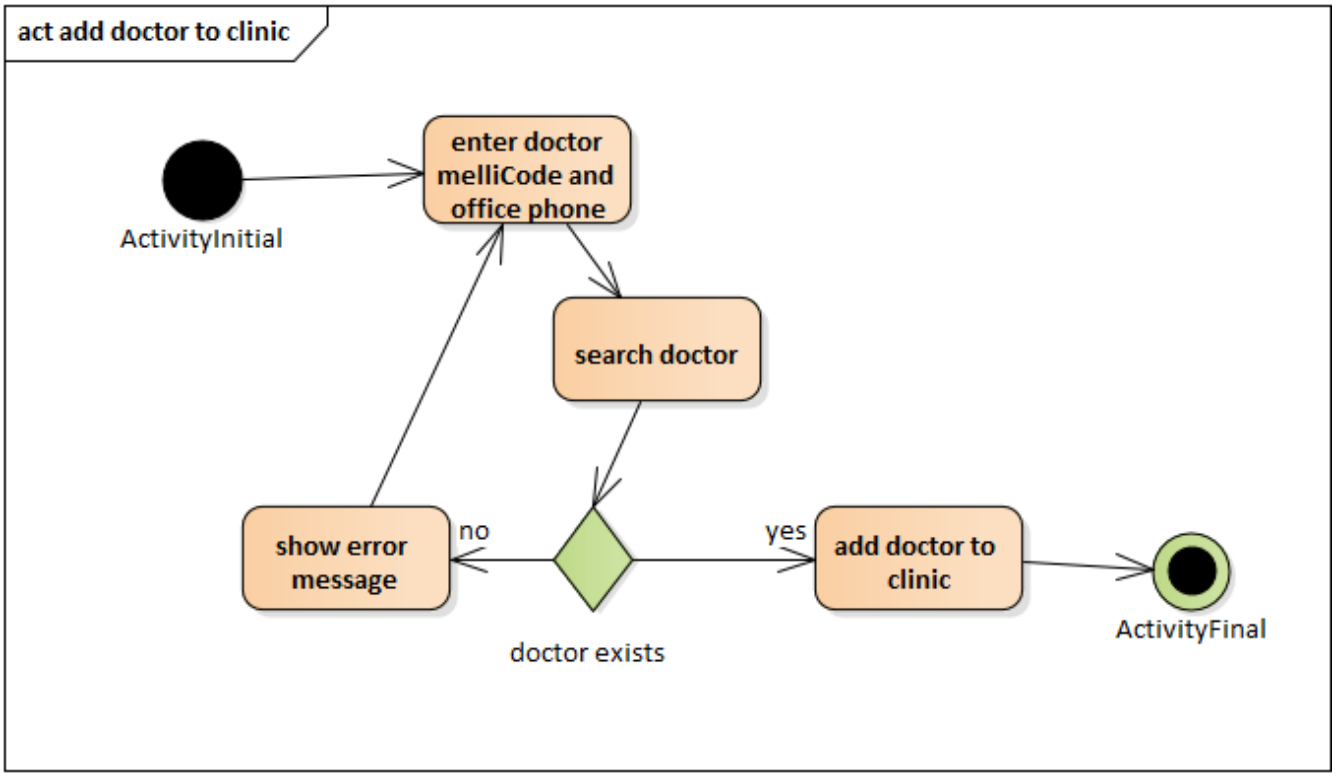
Use Case Number	۲۴
Use Case	جستجو بیماران توسط کلینیک
Summary	کلینیک میتواند از بین بیمارانی که به انها مراجعه کرده اند جستجو کنند و اطلاعات کاربری یا سابقه انها را مشاهده کنند
Actor	کلینیک
Trigger	دکمه بیماران من
Primary Scenario	کلینیک میتواند در بین بیمارانی که به او مراجعه کرده اند از طریق کدملی، جنسیت، نام خانوادگی، نوع درمان جستجو کند
Pre-Conditions	۱- کلینیک حداقل یک نوبت داشته باشد ۲- کلینیک به حساب کاربری خود وارد شده باشد



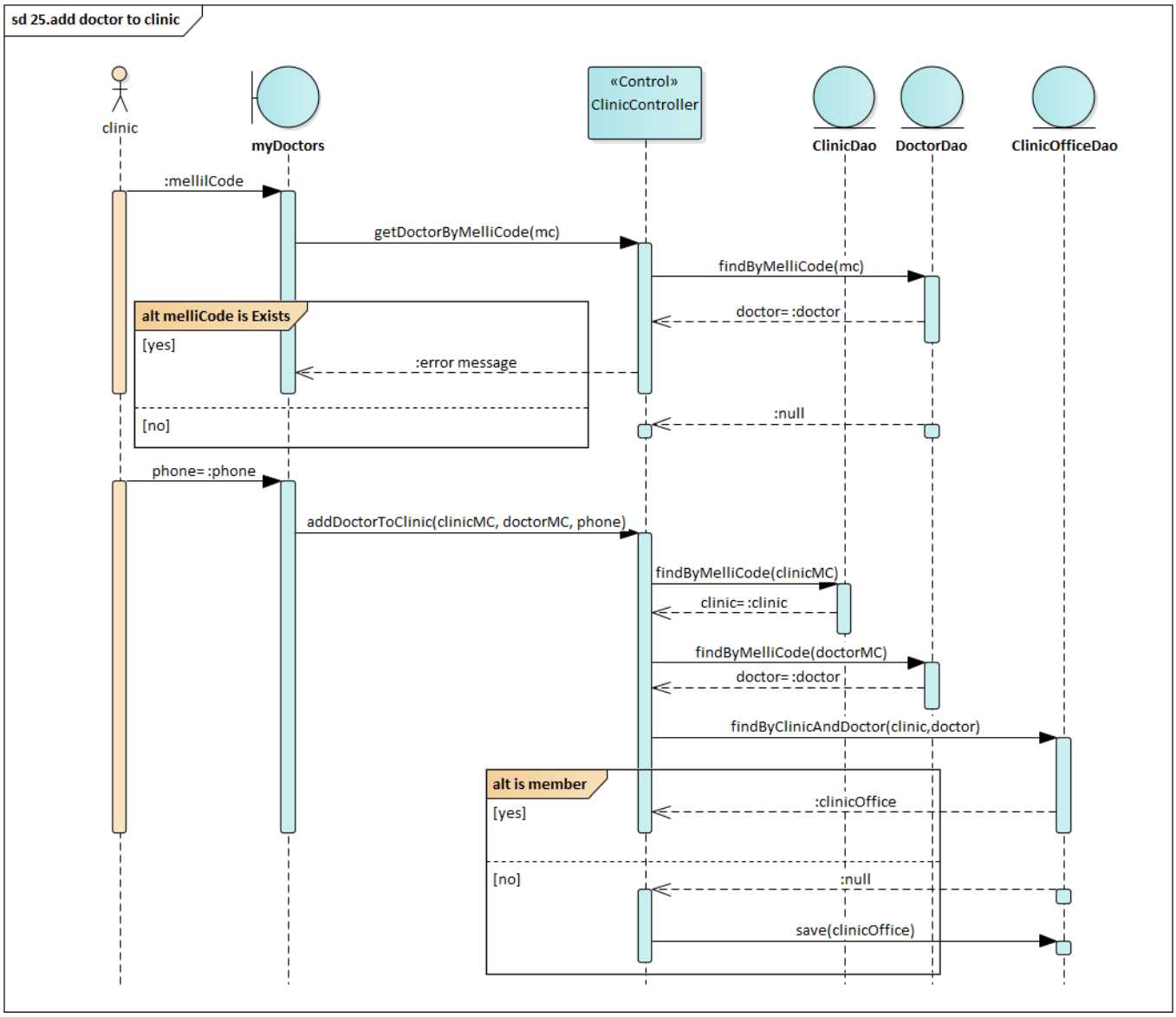
شکل ۳-۳۱ نمودار توالی جستجو بیماران کلینیک

جدول ۳-۲۵ مورد کاربرد اضافه کردن پزشک به کلینیک

Use Case Number	۲۵
Use Case	اضافه کردن دکتر به کلینیک
Summary	هر کلینیک میتواند به کلینیک خود پزشکانی را اضافه کند
Actor	کلینیک
Trigger	دکمه ثبت پزشک جدید در صفحه پزشکان
Primary Scenario	کاربر باید کدملی پزشک را جستجو کند به پزشک مورد نظر شماره تلفنی اختصاص دهد سپس پزشک را به کلینیک اضافه کند
Pre-Conditions	۱-کلینیک به حساب کاربری خود وارد شود ۲- دکتر باید از قبل ثبت نام کرده باشد



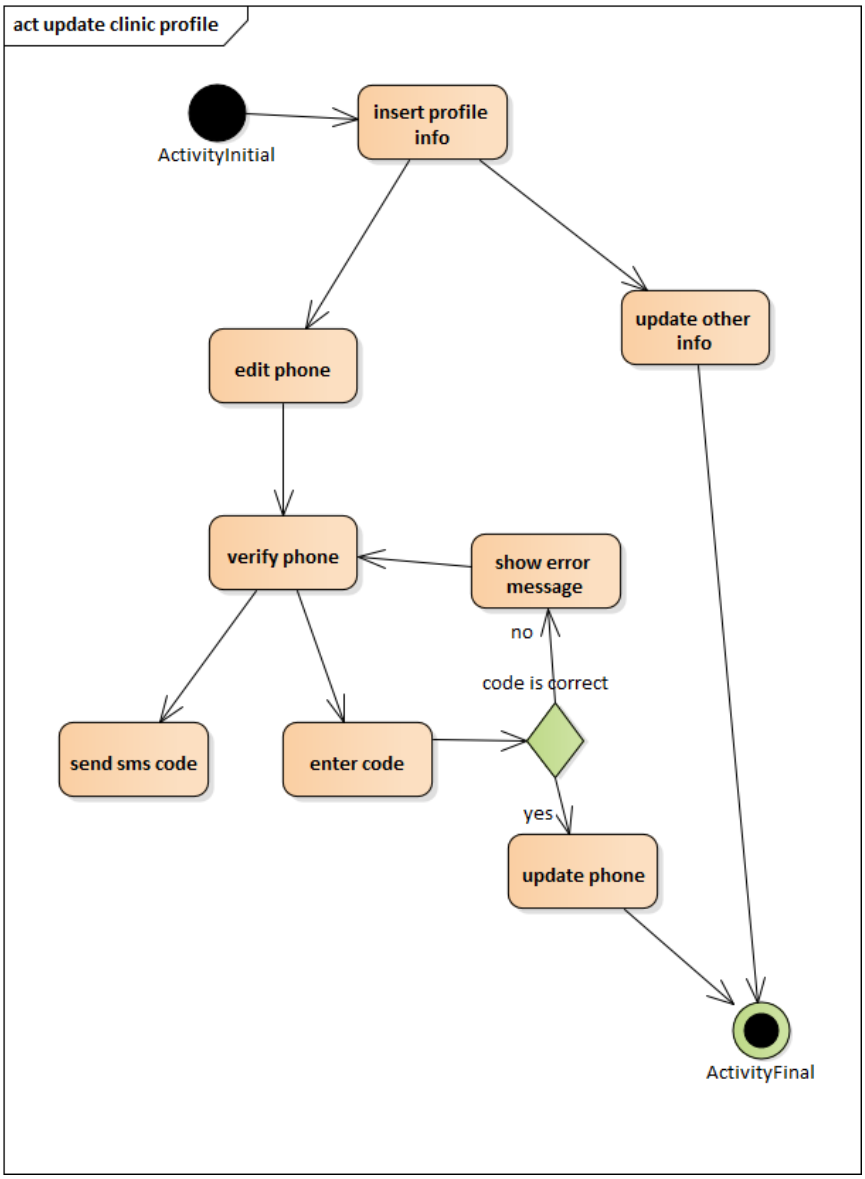
شکل ۳-۳۲ نمودار فعالیت اضافه کردن پزشک به کلینیک



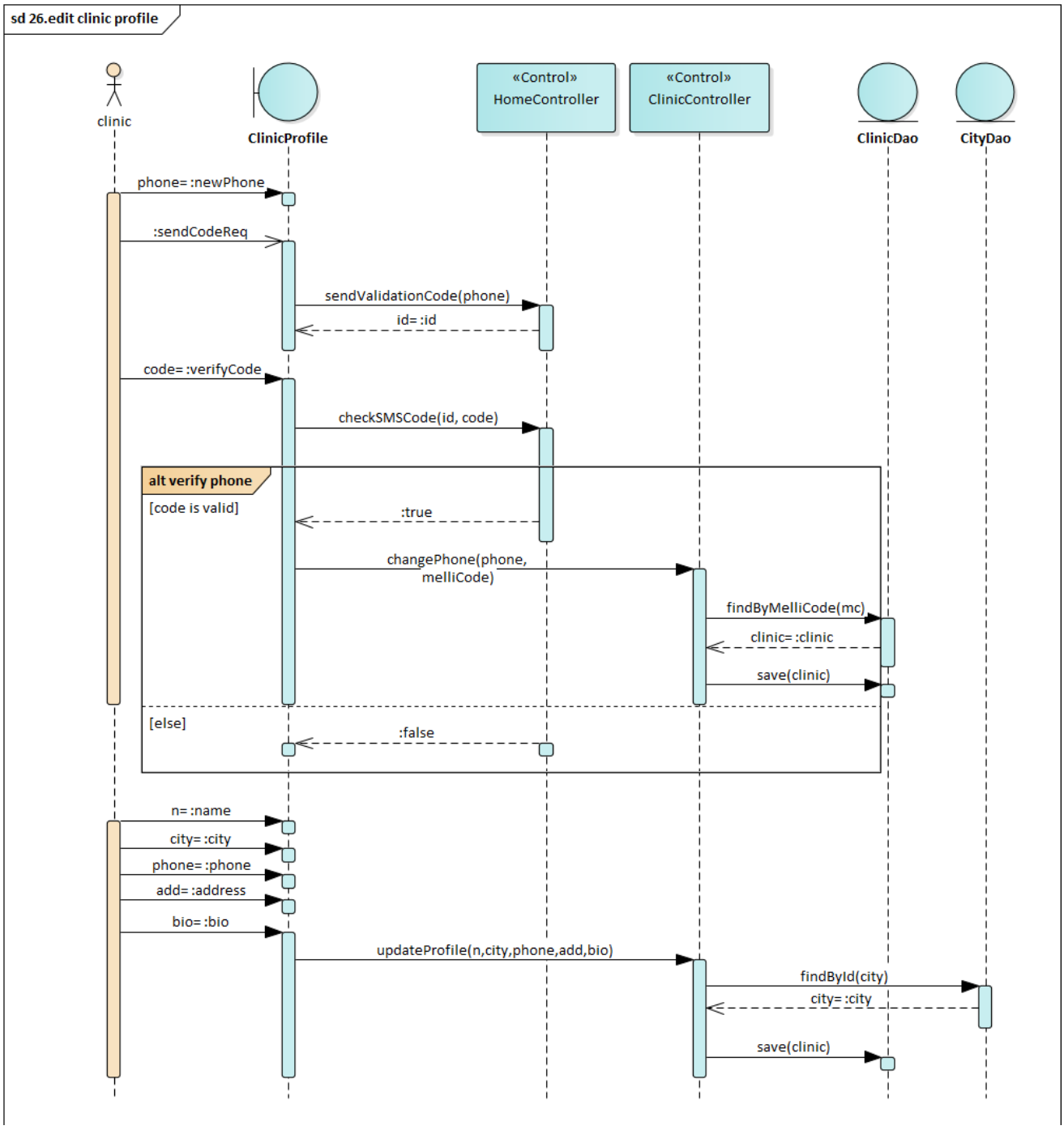
شکل ۳-۳۳ نمودار توالی اضافه کردن پزشک به کلینیک

جدول ۳-۲۶ مورد کاربرد تغییر اطلاعات کاربری کلینیک

Use Case Number	۲۶
Use Case	تغییر اطلاعات کاربری کلینیک
Summary	کلینکی میتواند اطلاعات کاربری خود را تغییر دهند
Actor	کلینیک
Trigger	دکمه پروفایل
Primary Scenario	کلینیک میتواند نام، شماره موبایل، شماره تلفن کلینیک، شهر خود را تغییر دهد
Pre-Conditions	کلینیک باید به حساب خود وارد شده باشد
Post-Conditions	۱- اگر شماره موبایل تکراری باشد تغییرات ثبت نخواهد شد



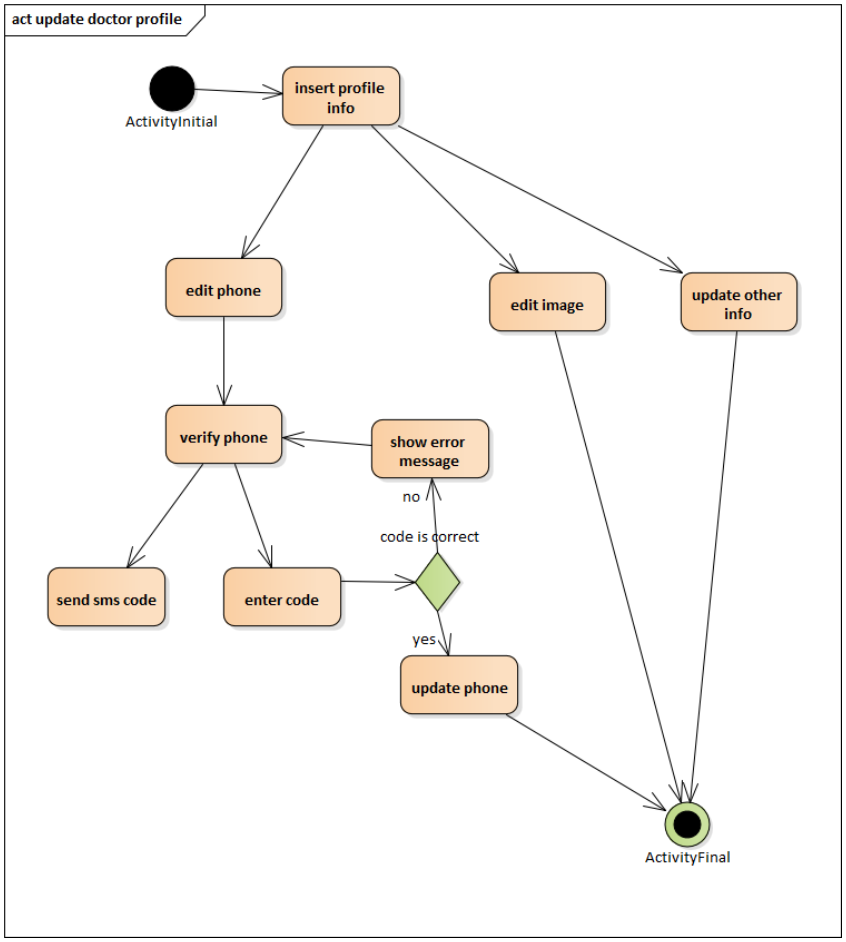
شکل ۳-۳۴ نمودار فعالیت تغییر اطلاعات کاربری کلینیک



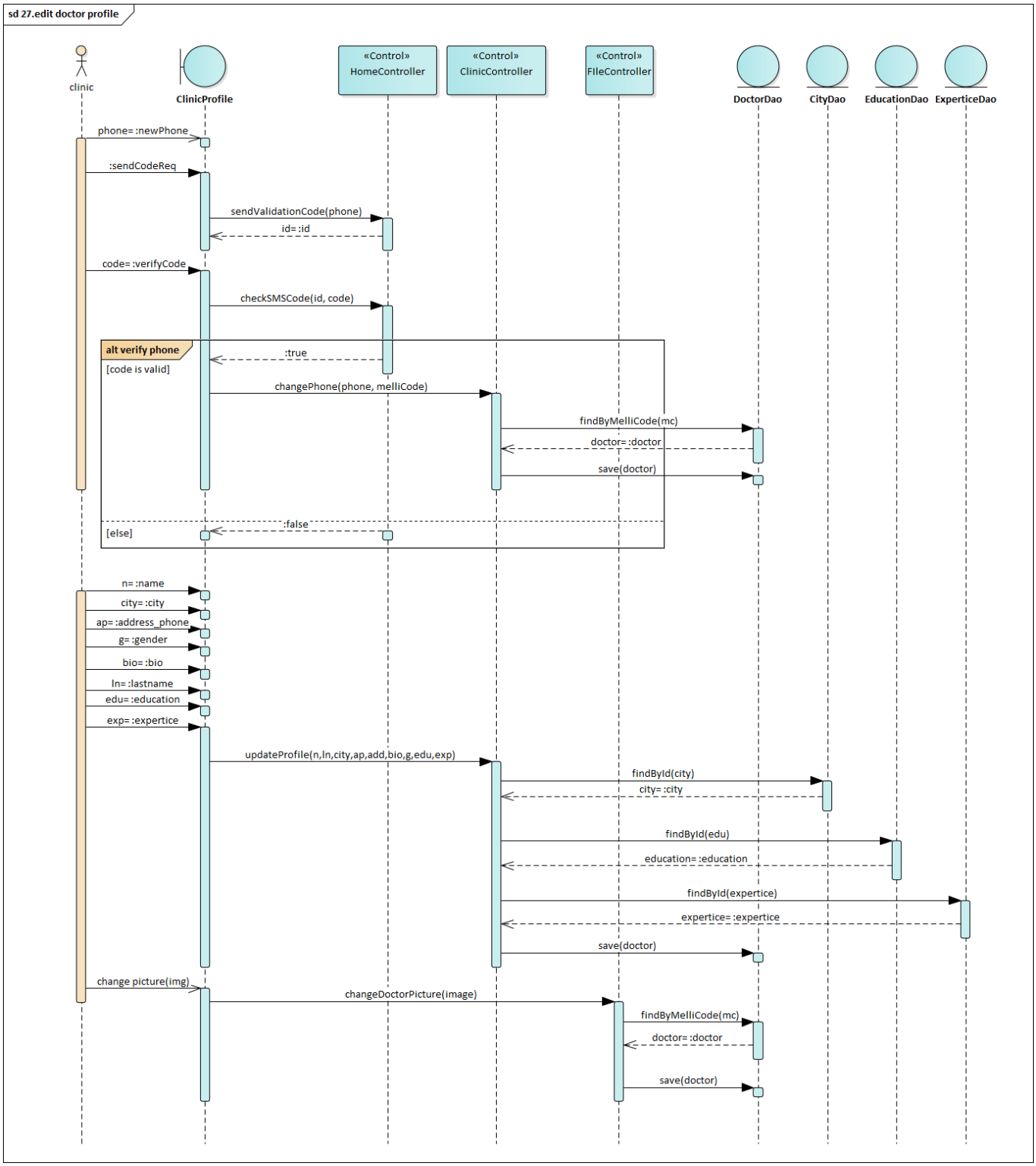
شکل ۳-۳۵ نمودار توالی تغییر اطلاعات کاربری کلینیک

جدول ۳-۲۷ مورد کاربرد تغییر اطلاعات کاربری پزشک

Use Case Number	۲۷
Use Case	تغییر اطلاعات کاربری پزشک
Summary	پزشک میتواند اطلاعات کاربری خود را تغییر دهد
Actor	پزشک
Trigger	دکمه پروفایل
Primary Scenario	پزشک میتواند نام، نام خانوادگی، شماره موبایل، شهر، تخصص، تحصیلات، جنسیت و بیوگرافی خود را تغییر دهد همچنین متواند ادرس و تلفن جدید برای خود اضافه یا حذف کند کلینیک هایی که پزشک را به خود اضافه کرده اند در پروفایل پزشک مشخص است و پزشک میتواند کلیک بر روی حذف از لیست پزشکان کلینیک خارج شود پزشک یک تصویر کاربری دارد که میتواند ان را تغییر دهد
Pre-Conditions	۱-پزشک باید به حساب کاربری خود وارد شده باشد
Post-Conditions	۱- اگر شماره تلفن تکراری باشد تلفن ثبت نخواهد شد



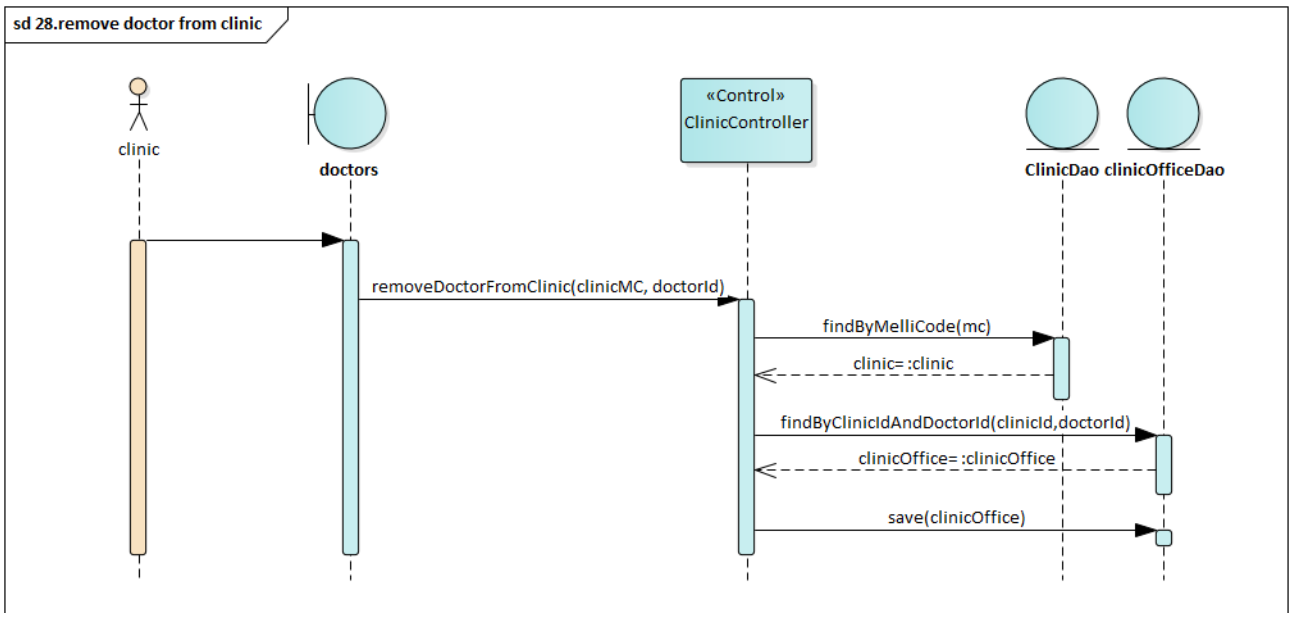
شکل ۳-۳۶ نمودار فعالیت تغییر اطلاعات کاربری پزشک



شکل ۳-۳۷ نمودار توالی تغییر اطلاعات کاربری پزشک

جدول ۳-۲۸ مورد کاربرد حذف پزشک از کلینیک

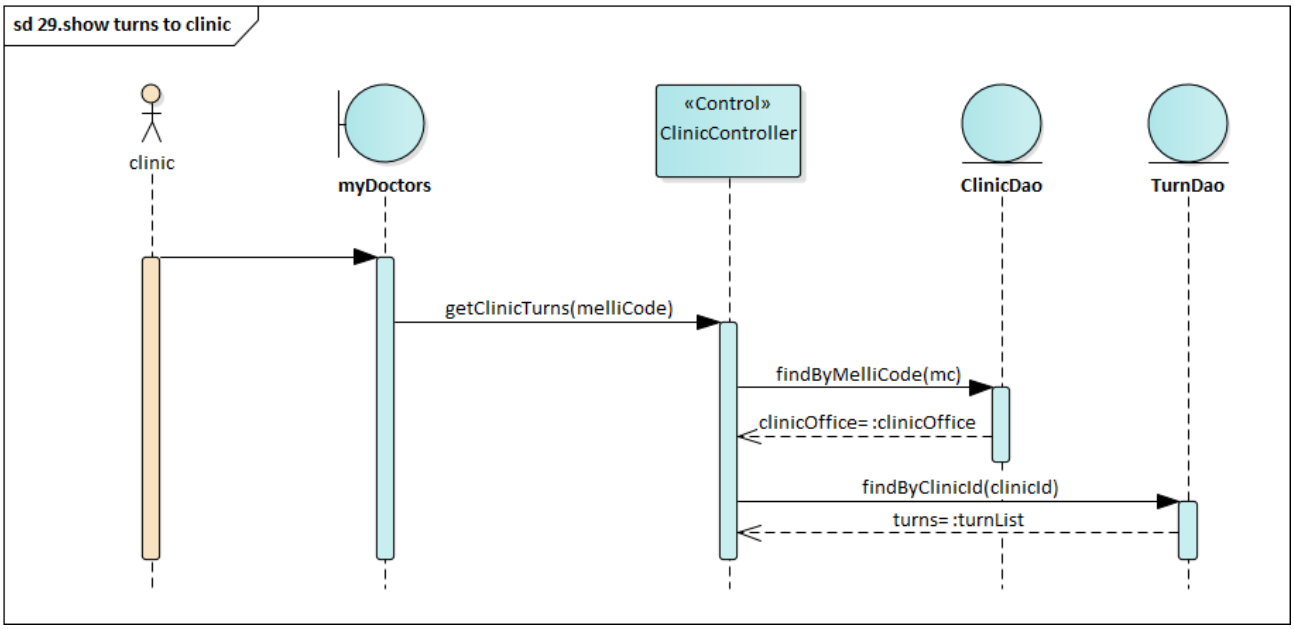
Use Case Number	۲۸
Use Case	حذف پزشک از کلینیک
Summary	کلینیک میتواند پزشکان خود را حذف کند
Actor	کلینیک
Trigger	دکمه حذف در صفحه پزشکان
Primary Scenario	هر پزشکی که به کلینیک اضافه میشود به در صفحه پزشکان کلینیک نمایش داده میشود و کلینیک می تواند هر پزشک را حذف کند که در این صورت از لیست پزشکان کلینیک خارج می شود
Pre-Conditions	کلینیک باید به حساب کاربری خود وارد شده باشد



شکل ۳-۳۸ نمودار توالی حذف پزشک از کلینیک

جدول ۳-۲۹ مورد کاربرد نوبت های کلینیک

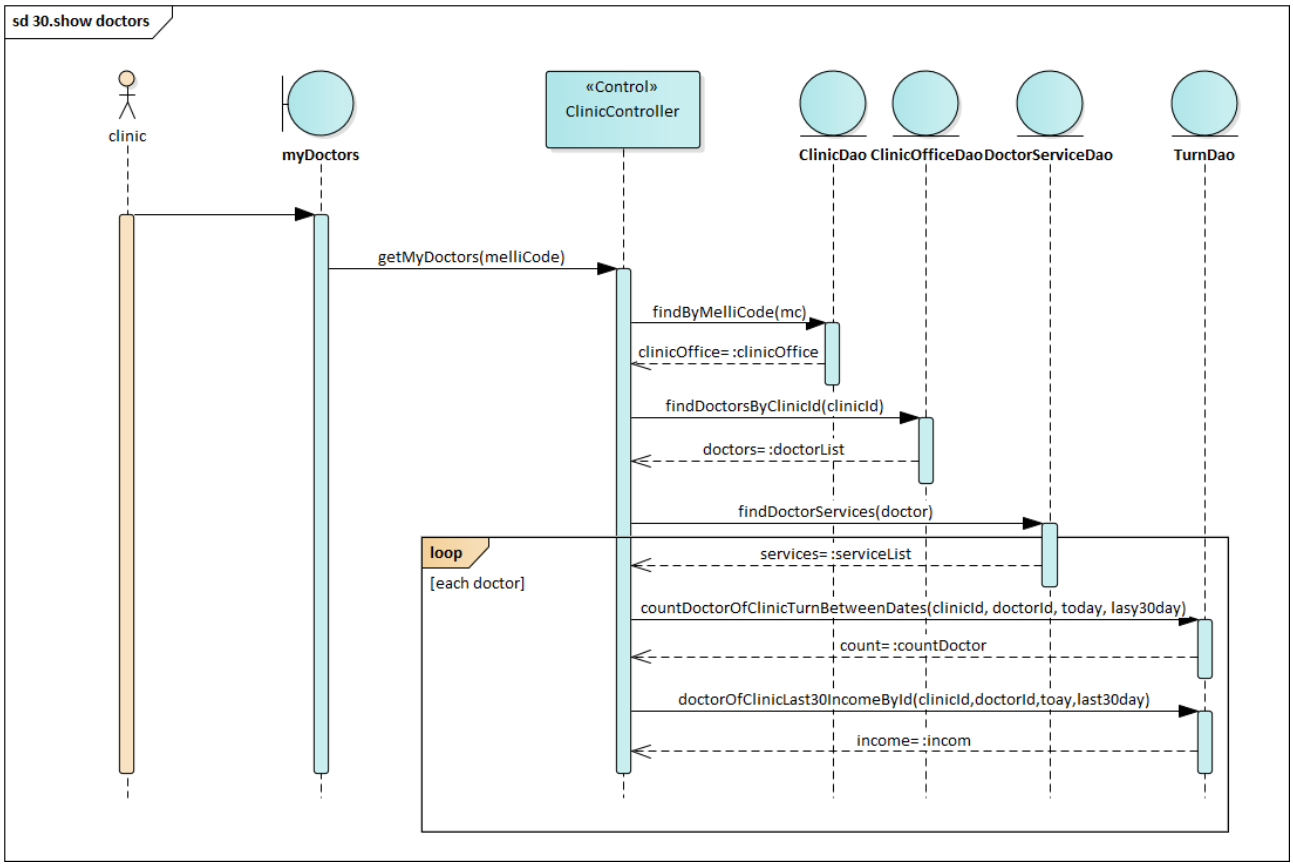
Use Case Number	۲۹
Use Case	نوبت ها کلینیک
Summary	کلینیک میتواند نوبت هایی که در پزشکان او ثبت میکند را ببیند
Actor	کلینیک
Trigger	دکمه نوبت ها
Primary Scenario	نوبت هایی که در ادرس ان ها کلینیک و توسط پزشکان کلینیک ثبت می شوند در قالب تقویم نمایش داده میشود
Pre-Conditions	۱- کلینیک پزشکی داشته باشد ۲- پزشکان کلینیک حداقل یک نوبت در کلینیک ثبت کرده باشند ۳- کلینیک به حساب کاربری خود وارد شده باشد



شکل ۳-۳۹ نمودار توالی نوبت های کلینیک

شکل ۳-۳۰ مورد کاربرد جستجو پزشکان توسط کلینیک

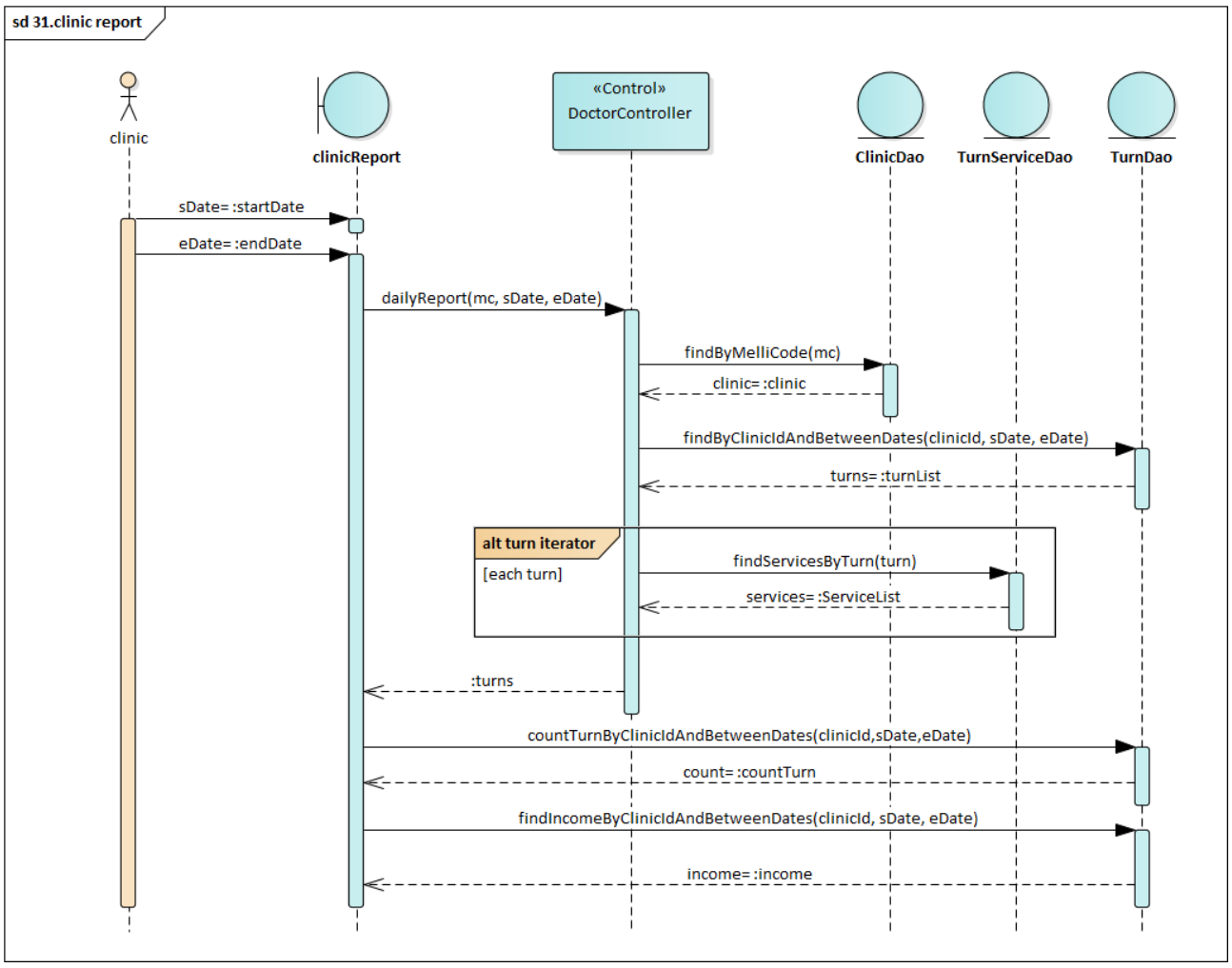
Use Case Number	۳۰
Use Case	جستجو پزشکان
Summary	کلینیک می تواند در بین پزشکان خود جستجو کند
Actor	کلینیک
Trigger	دکمه پزشکان
Primary Scenario	هر کلینیک میتواند از بین پزشکانی که به کلینیک اضافه کرده جستجو کند جستجو میتواند بر اساس نام خانوادگی پزشک، جنسیت، کدملی و شماره ملی صورت گیرد
Pre-Conditions	۱- کلینیک به حساب کاربری خود وارد شده باشد



شکل ۳-۴۰ نمودار توالی جستجو پزشکان توسط کلینیک

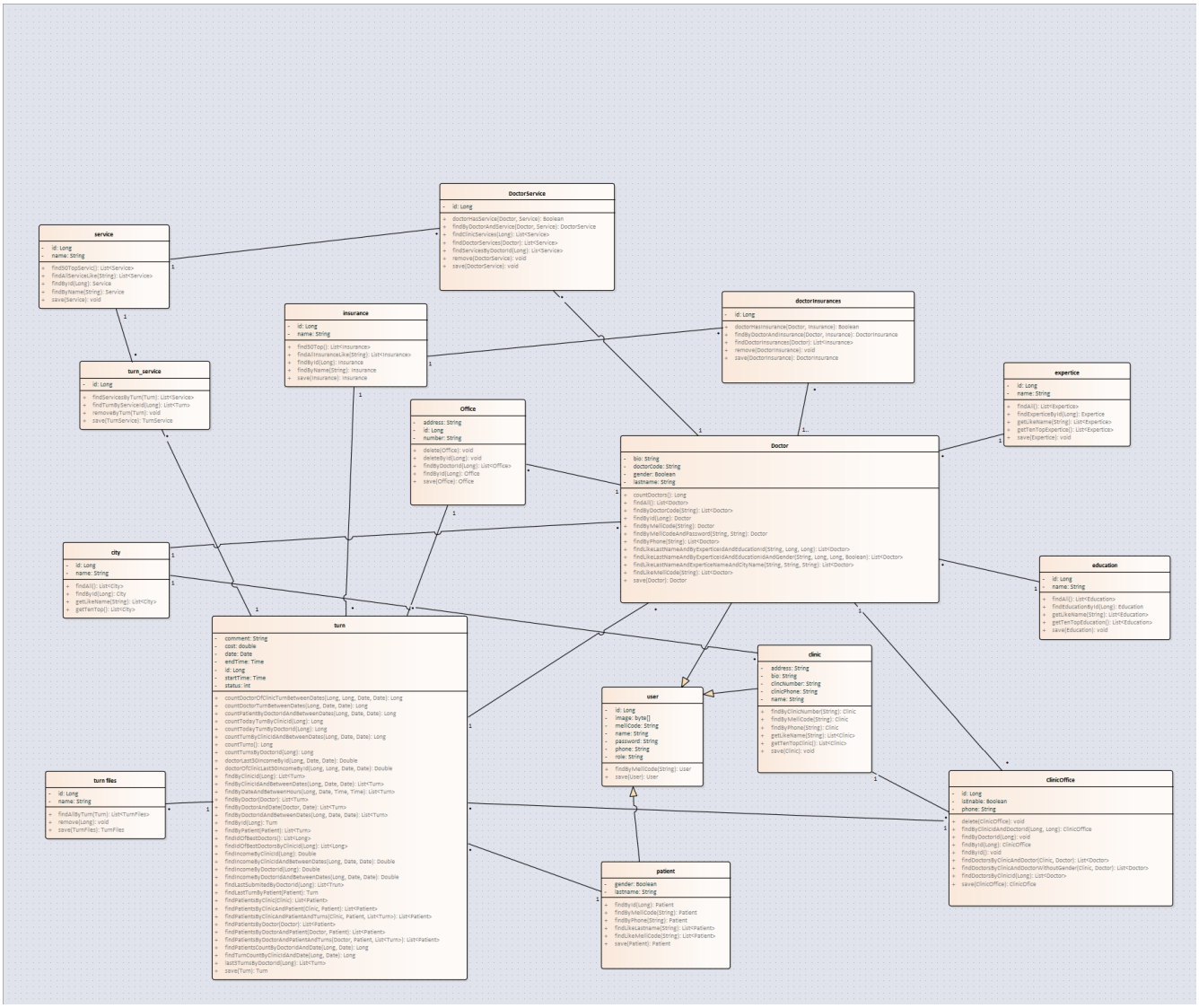
جدول ۳-۳۱ مورد کاربرد گزارش گیری توسط کلینیک

Use Case Number	۳۱
Use Case	گزارش گیری
Summary	هر کلینیک میتواند از نوبت هایی که پزشکان خود داشته اند گزارشی تهیه کند
Actor	کلینیک
Trigger	دکمه گزارش روزانه
Primary Scenario	کلینیک بازه زمانی را مشخص می کند و دران بازه چارتی که تعداد نوبت های کلینیک در هر تاریخ را نشان میدهد نمایش داده می شود به همراه جدولی که تاریخ، نام بیمار، جنیست، نوع درمان، نام پزشک، تخصص پزشک، وضعیت نوبت ومبلغ را برای هر نوبت نمایش می دهد
Pre-Conditions	۱- کلینیک به حساب کاربری خود وارد شده باشد



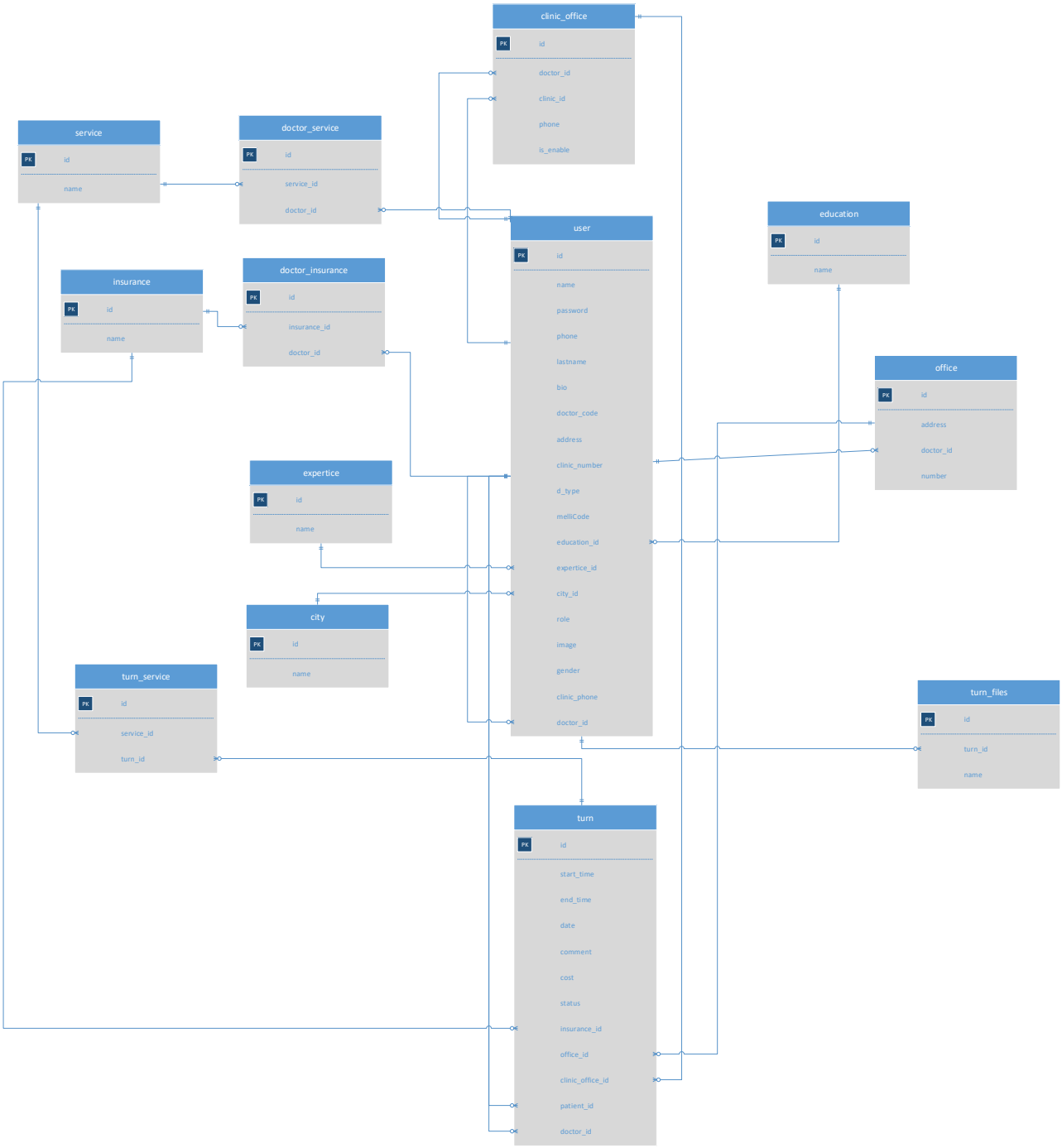
شکل ۳-۴۱ نمودار توالی گزارش گیری توسط کلینیک

دیاگرام کلاس (class diagram)



شکل ۳-۴۲ دیاگرام کلاس

دیاگرام موجودیت-رابطه (ERD)



شکل ۳-۴۳ دیاگرام موجودیت رابطه

فصل چهارم – شرح کد

کد های این سیستم به ۴ دسته تقسیم می شوند common, model, controller, view

View: بخش رابط کاربری این سیستم با react js نوشته شده است و بصورت وب سرویس با سرور ارتباط برقرار میکند

Controller: این سیستم دارای ۵ کنترلر می باشد.

PatientController, DoctorController,ClinicController, FileController این کنترل ها سطح دسترسی شان محدود شده و فقط نقش های مشخص شده میتوانند به این کنترلر ها دسترسی پیدا کنند

HomeController: این کنترلر دارای دسترسی ازاد می باشد و یوزکیس هایی چون ثبت نام کاربر و جستجو پزشکان و ... در ان پیاده سازی شده است

Model: کد های این بخش مربوط به ارتباط با پایگاه داده می باشد. همان طور که میدانید لایه مدل یکی از سنگین ترین قسمت های پروژه می باشد. در این سیستم ما از ORM هایبرنت استفاده کرده ایم

Common: در این قسمت کلاس های pojo پیاده سازی شده است

شرح عملیات احراز هویت و مجوز

همان طور که ذکر شد عملیات وارد شدن با استفاده از تکنیک JWT انجام میشود

```
public class JwtTokenRequest implements Serializable {

    private static final long serialVersionUID = -5616176897013108345L;

    private String username;
    private String password;

    public JwtTokenRequest() {
        super();
    }

    public JwtTokenRequest(String username, String password) {
        this.setUsername(username);
        this.setPassword(password);
    }
```

```
    public String getUsername() {
        return this.username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return this.password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

هر درخواست Jwt که به سرور ارسال میشود باید بدنه درخواست شامل username, password باشد تا توسط سرور پردازش شود در غیر این صورت درخواست رد می شود

```
public class JwtTokenResponse implements Serializable {

    private static final long serialVersionUID = 8317676219297719109L;

    private final String token;

    public JwtTokenResponse(String token) {
        this.token = token;
    }

    public String getToken() {
        return this.token;
    }
```

```
}
```

پاسخ هر درخواست JWT یک توکن می باشد

```
@RestController

@CrossOrigin

public class JwtAuthenticationRestController {


    @Value("${jwt.http.request.header}")

    private String tokenHeader;


    @Autowired

    private AuthenticationManager authenticationManager;


    @Autowired

    private JwtTokenUtil jwtTokenUtil;


    @Autowired

    private UserDetailsService jwtInMemoryUserDetailsService;


    @RequestMapping(value = "${jwt.get.token.uri}", method = RequestMethod.POST)

    public ResponseEntity<?> createAuthenticationToken(@RequestBody JwtTokenRequest
authenticationRequest)

            throws AuthenticationException {


        authenticate(authenticationRequest.getUsername(),
authenticationRequest.getPassword());


        final UserDetails userDetails = jwtInMemoryUserDetailsService

                .loadUserByUsername(authenticationRequest.getUsername());


        final String token = jwtTokenUtil.generateToken(userDetails);


        return ResponseEntity.ok(new JwtTokenResponse(token));
```

```
    }


    @RequestMapping(value = "${jwt.refresh.token.uri}", method = RequestMethod.GET)
    public ResponseEntity<?> refreshAndGetAuthenticationToken(HttpServletRequest request) {
        String authToken = request.getHeader(tokenHeader);
        final String token = authToken.substring(7);
        String username = jwtTokenUtil.getUsernameFromToken(token);
        JwtUserDetails user = (JwtUserDetails)
jwtInMemoryUserDetailsService.loadUserByUsername(username);

        if (jwtTokenUtil.canTokenBeRefreshed(token)) {
            String refreshedToken = jwtTokenUtil.refreshToken(token);
            return ResponseEntity.ok(new JwtTokenResponse(refreshedToken));
        } else {
            return ResponseEntity.badRequest().body(null);
        }
    }


    @ExceptionHandler({ AuthenticationException.class })
    public ResponseEntity<String> handleAuthenticationException(AuthenticationException e) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(e.getMessage());
    }


    private void authenticate(String username, String password) {
        Objects.requireNonNull(username);
        Objects.requireNonNull(password);

        try {
            authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(username, password));
        } catch (DisabledException e) {
            throw new AuthenticationException("USER_DISABLED", e);
        } catch (BadCredentialsException e) {
            throw new AuthenticationException("INVALID_CREDENTIALS", e);
```

```
        }
    }
}
```

این کلاس دو متد اصلی دارد

createAuthenticationToken() : درخواست هایی با متد POST که به URL:authenticate ارسال شوند را می پذیرد این درخواست ها username, password می باشند. بررسی میشود که نام کاربری با رمز عبور مطابقت داشته باشد اگر مطابقت نداشت پیام خطایی در پاسخ درخواست کاربر فرستاده میشود.

اگر نام کاربری و کلمه عبور با هم مطابقت داشت با نام کاربری درخواست دهنده توکنی ایجاد می شود و در پاسخ به او داده می شود

refreshAndGetAuthenticationToken : هر توکن تقریبا ۷ روز معتبر است این متد اعتبار توکن را تمدید میکند

```
@Component
public class JwtTokenAuthorizationOncePerRequestFilter extends OncePerRequestFilter {

    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    private UserDetailsService jwtInMemoryUserDetailsService;

    @Autowired
    private JwtTokenUtil jwtTokenUtil;

    @Value("${jwt.http.request.header}")
    private String tokenHeader;

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response, FilterChain chain)
            throws ServletException, IOException {
        logger.debug("Authentication Request For '{}'", request.getRequestURL());
```

```
        final String requestTokenHeader = request.getHeader(this.tokenHeader);


        String username = null;

        String jwtToken = null;

        if (requestTokenHeader != null && requestTokenHeader.startsWith("Bearer ")) {

            jwtToken = requestTokenHeader.substring(7);

            try {

                username = jwtTokenUtil.getUsernameFromToken(jwtToken);

            } catch (IllegalArgumentException e) {

                logger.error("JWT_TOKEN_UNABLE_TO_GET_USERNAME", e);

            } catch (ExpiredJwtException e) {

                logger.warn("JWT_TOKEN_EXPIRED", e);

            }

        } else {

            logger.warn("JWT_TOKEN_DOES_NOT_START_WITH_BEARER_STRING");

        }


        logger.debug("JWT_TOKEN_USERNAME_VALUE '{}'", username);

        if (username != null && SecurityContextHolder.getContext().getAuthentication() ==
null) {


            UserDetails userDetails =
this.jwtInMemoryUserDetailsService.loadUserByUsername(username);


            if (jwtTokenUtil.validateToken(jwtToken, userDetails)) {

                UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken =
new UsernamePasswordAuthenticationToken(

                        userDetails, null, userDetails.getAuthorities());

                usernamePasswordAuthenticationToken

                        .setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));


SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

            }

        }
```

```
        chain.doFilter(request, response);
    }
}
```

اگر کاربر به هر URL درخواست ارسال کند ابتدا از متد doFilterInternal عبور میکند مگر برخی URL هایی که مشخص میکنیم این متد بررسی میکند که کاربر توکن داشته باشد. سپس از توکن کاربر username او را استخراج میکند سپس اگر توکن کاربر معتبر باشد و هنوز احراز هویت نشده باشد هویتش را احراز میکند

```
@Configuration
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true, securedEnabled = true)
public class JWTWebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private JwtUnAuthorizedResponseAuthenticationEntryPoint
jwtUnAuthorizedResponseAuthenticationEntryPoint;

    @Autowired
    private UserDetailsService jwtInMemoryUserDetailsService;

    @Autowired
    private JwtTokenAuthorizationOncePerRequestFilter jwtAuthenticationTokenFilter;

    @Value("${jwt.get.token.uri}")
    private String authenticationPath;

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.userDetailsService(jwtInMemoryUserDetailsService).passwordEncoder(passwordEncoderBean()
);

    }
```

```
    @Bean
    public PasswordEncoder passwordEncoderBean() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    @Override
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Override
    protected void configure(HttpSecurity httpSecurity) throws Exception {
        httpSecurity.csrf().disable().exceptionHandling()
.authenticationEntryPoint(jwtUnAuthorizedResponseAuthenticationEntryPoint).and()
                .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS).and()
                .authorizeRequests()
                .anyRequest().authenticated();

        httpSecurity.addFilterBefore(jwtAuthenticationTokenFilter,
UsernamePasswordAuthenticationFilter.class);

        httpSecurity.headers().frameOptions().sameOrigin() // H2 Console Needs this setting
                .cacheControl(); // disable caching
    }

    @Override
    public void configure(WebSecurity webSecurity) throws Exception {
        webSecurity.ignoring()
                .antMatchers(HttpMethod.POST, authenticationPath)
```

```
                .antMatchers(HttpMethod.OPTIONS, "/**")
                .antMatchers(HttpMethod.GET, "/") // Other Stuff You want to Ignore
                .antMatchers( "/api/*")
                .antMatchers( "/temp")
                .antMatchers( "/temp/**", "/temp/*", "/", "/**", "/*");// for access to
static folder


    }


}
```

این کلاس که در واقع کلاسی برای پیکربندی امنیتی می باشد دو متد مهم دارد

configure(HttpSecurity httpSecurity) : در این متد فیلتری که در کلاس قبلی ایجاد کردیم را قبل از هر درخواستی اعمال میکنیم

configure(WebSecurity webSecurity) : دراین متد url هایی که هر کاربری میتواند به انها درخواست بفرستد را تعیین میکنیم تا فیلتر امنیتی را روی انها اعمال نکند

فرایند ثبت نام پزشک

```
@Component
@Entity
@Scope(scopeName = "request")
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "d_type", discriminatorType = DiscriminatorType.STRING)
public class Users{
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "userSeqGen")
    @SequenceGenerator(name = "userSeqGen", sequenceName = "USER_SEQ_GEN", allocationSize =
1)
    protected Long id;
    protected String melliCode;
    @Column(name = "name")
    protected String name;
```

```
    protected String password;

    protected String phone;

    protected String role;

    protected byte[] image;


    public byte[] getImage() {

        return image;

    }


    public void setImage(byte[] image) {

        this.image = image;

    }


    public Users() {

    }


    public Users(Long id, String melliCode, String name, String password, String phone,
String role) {

        this.id = id;

        this.melliCode = melliCode;

        this.name = name;

        this.password = password;

        this.phone = phone;

        this.role = role;

    }


    public String getRole() {

        return role;

    }


    public void setRole(String role) {

        this.role = role;

    }
```

```
public String getPassword() {
    return password;
}

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getMelliCode() {
    return melliCode;
}

public void setMelliCode(String melliCode) {
    this.melliCode = melliCode;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public void setPassword(String password) {
    this.password = password;
}
```

```
    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

}
```

این سیستم دارای ۳ کاربر می باشد که هر ۳ از کلاس User ارث می برند

@Inheritance(strategy = InheritanceType.***SINGLE_TABLE***) این انوتیشن استراتژی ذخیره داده های این کلاس و کلاس های فرزند در پایگاه داده را مشخص میکند

در اینجا از استراتژی Single Table استفاده شده است

بر طبق این استراتژی تمامی کلاس های فرزند به یک جدول نگاشت می شوند جدولی بنام user این جدول به اندازه مجموع خصیصه های این سه کلاس ستون دارد.

@DiscriminatorColumn(name = **"d_type"**, discriminatorType = DiscriminatorType.***STRING***)

از انجایی که تمام کلاس های به یک جدول نگاشت میشوند ستونی را در این جدول به نوع داده اختصاص میدهیم که با این انوتیشن نام ستون را مشخص میکنیم. برای مثال اگر بخواهیم داده های کلاس doctor را ذخیره کنیم ستون d_type با مقدار doctor پر خواهد شد

```
@Component
@Entity
@Scope(scopeName = "request")
@DiscriminatorValue("doctor")
public class Doctor extends Users implements Cloneable {

    String doctorCode;
    @Column(name = "lastname")
    String lastname;
    String bio;
    Boolean gender;
```

```
@ManyToOne( optional = false)
@JoinColumn(name = "education_id")
Education education;

@OneToMany(mappedBy = "doctor")
List<DoctorInsurance> insuranceList;

@OneToMany
List<DoctorService> serviceList;

@OneToMany(mappedBy = "doctor")
List<Turn> turnList;

@ManyToOne(optional = false)
@JoinColumn(name = "city_id")
City city;

@OneToMany(mappedBy = "doctor")
List<Office> officeList;

@OneToMany(mappedBy = "doctor")
List<ClinicOffice> clinics;

@ManyToOne(optional = false)
@JoinColumn(name = "expertice_id")
Expertice expertice;

@OneToMany(mappedBy = "doctor")
List<Patient> myPatients;


public List<Turn> getTurnList() {
```

```
    return turnList;
}

public void setTurnList(List<Turn> turnList) {
    this.turnList = turnList;
}

public List<Patient> getMyPatients() {
    return myPatients;
}

public void setMyPatients(List<Patient> myPatients) {
    this.myPatients = myPatients;
}

public Boolean getGender() {
    return gender;
}

public void setGender(Boolean gender) {
    this.gender = gender;
}

public List<ClinicOffice> getClinics() {
    return clinics;
}

public void setClinics(List<ClinicOffice> clinics) {
    this.clinics = clinics;
}

public List<Office> getOfficeList() {
    return officeList;
```

```
}

public void setOfficeList(List<Office> officeList) {
    this.officeList = officeList;
}

public City getCity() {
    return city;
}

public void setCity(City city) {
    this.city = city;
}

public Expertice getExpertice() {
    return expertice;
}

public void setExpertice(Expertice expertice) {
    this.expertice = expertice;
}




public List<DoctorService> getServiceList() {
    return serviceList;
}

public void setServiceList(List<DoctorService> serviceList) {
    this.serviceList = serviceList;
}

public List<DoctorInsurance> getInsuranceList() {
```

```
    return insuranceList;
}

public void setInsuranceList(List<DoctorInsurance> insuranceList) {
    this.insuranceList = insuranceList;
}

public Education getEducation() {
    return education;
}

public Doctor setEducation(Education education) {
    this.education = education;
    return this;
}

public String getDoctorCode() {
    return doctorCode;
}

public Doctor setDoctorCode(String doctorCode) {
    this.doctorCode = doctorCode;
    return this;
}

public String getLastname() {
    return lastname;
}

public Doctor setLastname(String lastname) {
    this.lastname = lastname;
    return this;
}
```

```
    public String getBio() {
        return bio;
    }

    public Doctor setBio(String bio) {
        this.bio = bio;
        return this;
    }

}
```

@OneToMany, @ManyToOne جز هایبرنت می باشند. درصورتی که داخلی کلاسی نیاز به استفاده از کلاس های دیگر داشته باشیم برای مثلا هر دکتر یک شهر دارد باید در ابتدا تعریف ان کلاس این انوتیشن ها گذاشته شود این انوتیشن ها در واقع پیاده ساز روابط یک به چند و چند به یک جداول رابطه ای هستند

نکته مهمی که راجع به @ManyToOne وجود دارد این است که این انوتیشن eager می باشد.

روابطی که بصورت eager هستند هنگام واکشی اطلاعات از پایگاه داده اطلاعات کلاس های داخلش را نیز واکشی میکند.

```
@PostMapping(value = "/registerDoctor")
public Boolean registerDoctor(@RequestBody BaseJsonNode json){
    Clinic clinic = new Clinic();



    Doctor doctor = new Doctor();
    doctor.setName(json.get("name").asText());
    doctor.setLastname(json.get("lastname").asText());
    doctor.setMelliCode(json.get("melliCode").asText());
    String pass = DataConfiguration.encode_pass(json.get("Password").asText());
    doctor.setPassword(pass);
    doctor.setPhone(json.get("phone").asText());
    doctor.setDoctorCode(json.get("doctorCode").asText());
```

```
doctor.setEducation(new Education().setId(json.get("education").asLong()));
doctor.setExpertice(new Expertice().setId(json.get("expertice").asLong()));
doctor.setCity(new City().setId(json.get("city").asLong()));
String directory = DataConfiguration.getResources_path()+"images\\male-avatar.png";
File file = new File(directory);
byte[] bFile = new byte[(int) file.length()];

try {
    FileInputStream fileInputStream = new FileInputStream(file);
    fileInputStream.read(bFile);
    fileInputStream.close();
} catch (Exception e) {
    e.printStackTrace();
}

doctor.setImage(bFile);
doctor.setRole("ROLE_DOCTOR");
doctor.setGender(json.get("gender").asBoolean());
Doctor savedDoctor = doctorDao.save(doctor);

String address = json.get("address").asText();
if(!(address.equals(""))){
    Office office = new Office();
    office.setDoctor(savedDoctor);
    office.setAddress(address);
    officeDao.save(office);
}

if(json.get("clinicNumber").asText().equals(""))
    return true;
else{
    clinic.setId(json.get("clinicNumber").asLong());
    ClinicOffice clinicOffice = new ClinicOffice();
```

```
        clinicOffice.setClinic(clinic);
        clinicOffice.setDoctor(savedDoctor);
        clinicOfficeDao.save(clinicOffice);
    }
    return true;

}
```

این متد قسمتی از کلاس HomeController می باشد که درخواست های POST که به /registerDoctor فرستاده میشود را می پذیرد. اطلاعات پزشک بصورت JSON فرستاده میشود و پس از استخراج انها شئ جدید از کلاس پزشک می سازد و ان را به doctorDao در لایه مدل می فرستد تا ذخیره کند

```
@Transactional
public Doctor save(Doctor doctor){
    return entityManager.merge(doctor);
}
```

این متد بخشی از کلاس DoctorDao می باشد

@Transactioanl تضمین می کند که این قطعه کد در قالب یک تراکنش اجرا شود یعنی خواص ACID را داشته باشد

در جاوا با واسط JDBC به پایگاه داده متصل میشویم ولی ما از ORM هایبرنت استفاده کرده ایم. هایبرنت واسطی دارد که روی JDBC می نشیند که entityManager نام دارد. تفاوت انها در این است که JDBC با جدول و رابطه سروکار دارد ولی entityManager با شئ و شئ گرایی در این قسمت شئ از جنس Doctor که از لایه کنترلر به این لایه فرستاده شده بود را ذخیره کرده ایم

دانلود فایل از سرور

زمانی که کاربر روی فایل پیوست شده در هر نوبت کلیک میکند نام فایل و ایدی نوبت به کنترلر FileController و به متد زیر ارسال میشود

```
@PostMapping(value = "/getTurnFile")
public String  getFilesByTurn(@RequestBody BaseJsonNode json){
    Long turnId = json.get("turnId").asLong();
    String fileName = json.get("fileName").asText();

    String directoryName = "";
    boolean check = true;
    String rand="";
    while (check){
        rand = "tabib";
        for(int i=0; i < 20; i++){
            int random = new Random().nextInt(9);
            rand += String.valueOf(random);
        }

        directoryName = DataConfiguration.getStatic_path()+"temp\\"+rand+"\\";
        File directory = new File(directoryName);
        if ( directory.exists()){
            continue;
        }

        directory.mkdirs();
```

```
        check = false;
    }

    Path sourceDirectory =
Paths.get(DataConfiguration.getUploads_path()+turnId+"\\"+fileName);
    Path targetDirectory = Paths.get(directoryName+fileName);

    //copy source to target using Files Class
    try {
        Files.copy(sourceDirectory, targetDirectory);
    } catch (IOException e) {
        e.printStackTrace();
    }

    Timer timer = new Timer();
    String finalDirectoryName = directoryName;
    TimerTask ts = new TimerTask() {
        @Override
        public void run() {
            System.out.println("file removed...");
            File f = new File(finalDirectoryName);
            if(f.exists()){
                String[]entries = f.list();
                for(String s: entries){
                    File currentFile = new File(f.getPath(),s);
                    currentFile.delete();
                }
                f.delete();
            }
            timer.cancel();
        }
    };
    timer.schedule(ts, 1800000, 1);//30 minues
```

```
    return DataConfiguration.domain+"temp/"+rand+"/"+fileName;
}
```

یک مسیر تصادفی ۲۵ حرفی ایجاد میشود و دایرکتوری با این نام ایجاد میشود و فایل مربوطه به از دایرکتوری فایل های اپلود شده به دایرکتوری جدید کپی میشود و ادرس ان به کاربر داده میشود

مسیر ایجاد شده تنها ۳۰ دقیقه معتبر می باشد و بعد از ان مسیر از بین میرود

مراجع

[1] "java-wikipedia," 2019. [Online]. Available: https://fa.wikipedia.org/wiki/جاوا_(زبان_برنامه‌نویسی).

[2] "wikipedia spring framework," 2019. [Online]. Available: https://fa.wikipedia.org/wiki/اسپرینگ_فریمورک.

[3] "wikipedia hibernate," 2019. [Online]. Available: https://fa.wikipedia.org/wiki/هایبرنیت_(جاوا).

[4] "rest full api," 2019. [Online]. Available: https://sokanacademy.com.

[5] "react js," [Online]. Available: https://fa.wikipedia.org/wiki/ReactJS.

[6] "virgool," [Online]. Available: https://virgool.io.

[۷] ا. کازرونی و ا. مهرزادگان, "اهمیت یکپارچگی و ارکان آن در سازمان," ۱۳۹۳.