# SOS GAME

**تکین جزایری، 9813006**

برنامهٔ SOS به صورت 5 تابع اصلی تعریف شده است. در ابتدا سه ماژول sqlite3 برای کار با Database، tkinter برا ایجاد GUI و random برای تعیین تصادفی شروع کنندهٔ بازی import شده اند. در اجرای این برنامه از یک database به نام User_Info.db استفاده شده است که شامل دو table با نام های admin_info و user_info است که admin_info تنها شامل رمز ادمین می باشد و user_info شامل اطلاعات بازیکنان در شش ستون (user، pass، fName، lName، gameNum و winNum) می باشد. 5 تابع این برنامه شامل موارد زیر است:

1) برنامه با تابع start شروع می شود. این تابع ابتدا از طریق sqlite3، رمز ادمین و همچنین اطلاعات بازیکنان را از User_Info.db به صورت رشتهٔ adminPass و لیست record دریافت می کند. سپس صفحهٔ startWindow با استفاده از tkinter تعریف می شود که شامل دو label و دو entry برای وارد کردن username و password کاربر است. همچنین شامل دو button است که به ترتیب توابع signIn و signUp را اجرا می کنند و برای ورود کاربر و ایجاد حساب کاربری جدید به کار می روند. در صورتی که اطلاعات وارد شده در label ها مربوط به ادمین باشد، با اجرای تابع signIn، تابع admin اجرا می شود و در صورتی که اطلاعات مربوط به یکی از کاربرها باشد، صفحهٔ مربوط به کاربر مورد نظر با فراخواندن تابع dashboard در دسترس قرار می گیرد. اگر اطلاعات این قسمت نادرست باشد، ضمن پاک کردن فیلدها به کاربر ارور داده می شود. تابع signUp نیز با ارجاع به تابع addUser، کاربر را به صفحهٔ ایجاد حساب کاربری وارد می کند.
2) تابع dashboard با دریافت info که لیستی شامل اطلاعات یک کاربر است، اجرا می شود. این تابع صفحهٔ dashboardWindow را شامل اطلاعات کاربر به صورت یک label در بالای صفحه و پنج button زیر آن است. با فشردن Start new game، تابع secondSignIn اجرا می شود که مشابه با تابع start اطلاعات کاربر دوم را دریافت می کند و علاوه بر آن، عددی به عنوان تعداد سطرها و ستون ها باید وارد شود که باید بیش از 3 باشد و در غیر اینصورت، ارور ظاهر می شود. اگر مشکلی در اطلاعات وارد شده نباشد، تابع play اجرا می شود. Edit information button، تابع infoEdit را اجرا می کند که در یک صفحهٔ جدید، نام و نام خانوادگی مورد نظر کاربر را دریافت می کند و با فشردن Confirm، از طریق تابع edit نام و نام خانوادگی کاربر را تغییر داده و سپس مجدداً اطلاعات تمامی کاربران را دریافت کرده و با اجرای مجدد تابع dashboard، کاربر دو مرتبه به dashboardWindow برگرداننده می شود. تابع passChange با فشردن Change password اجزا می شود که در یک صفحهٔ جدید با دریافت password قدیمی، password جدید و تکرار password جدید، پس از فشردن Confirm button و اجرای change، اگر password قدیمی مطابق با password کاربر نباشد و یا password جدید و تکرار آن مشابه نباشند، ارور می دهد و فیلدهایی که به اشتباه پر شده اند، پاک می شوند. اگر این اشتباهات پیش نیامده باشد، رمز جدید در database به روز شده و کاربر به dashboard بر می گردد. Exit نیز با destroy کردن dashboardWindow و اجرای start، کاربر از صفحهٔ کاربری اش خارج می کند. آخرین button، Sign Out است که در تابع signOut، پس از پیامی که برای اطمینان از خروج برای کاربر فرستاده می شود، در صورت تأیید کاربر، اطلاعات کاربر از database حذف شده و کاربر به startWindow فرستاده می شود.
3) سومین تابع، admin است که به عنوان ورودی، اطلاعات کاربران و رمز ادمین را دریافت می کند. با اجرای تابع، صفحهٔ adminWindow ایجاد می شود که شامل یک listbox از کاربران همراه با scrollbar و سه button می باشد. button اوّل، Add a user است که ادمین را به تابع addUser ارجاع می دهد. همچنین با فشردن Change password، در تابع passChange مشابه آنچه در تابع admin برای تغییر password انجام می شود، رخ می دهد؛ با این تفاوت که اوّلاً در این تابع تغییر رمز با تغییر admin_info table در database انجام می شود، ثانیاً در آخر برنامه ادمین به adminWindow برگردانده

می شود و ثالثاً اگر ادمین، password خود را "12356" قرار دهد، برنامه قبول نمی کنند و ارور می دهد. در هنگام اجرای تابع admin نیز در صورتی که password "123456" باشد، در ابتدا تابع passChange اجرا می شود. آخرین button نیز ادمین را از adminWindow خارج می کند و start را اجرا می کند. در lisbox، با کلیک روی هر یک از موارد، تابع accountManagement اجرا می شود که شامل صفحهٔ AMWindow می باشد و این صفحه متشکّل از اطلاعات کاربر مورد نظر و همچنین سه button است که به ترتیب توابع editAccount، deleteAccount و exitAM است که مشابه توابع infoEdit، signOut و exitD در تابع dashboard هستند با این تفاوت که در آنها در آخر تابع، به جای dashboard، admin اجرا می شود.

4) تابع addUser دو پارامتر isAdmin و record را دریافت می کند که پارامتر اوّل، با عدد 0 یا 1 نشان می دهد که ورود از سمت کاربر یا ادمین انجام شده است و پارامتر دوم، لیستی شامل اطلاعات کاربران است. در این تابع، صفحهٔ addUserWindow ایجاد می شود که شامل پنج label و entry برای username، نام، نام خانوادگی، password و تکرار password می باشد. همچنین Confirm button در صفحه ایجاد می شود که تابع enterInfo را اجرا می کند. تابع enterInfo ابتدا username، password و تکرار password وارد شده را دریافت می کند و سپس username را با username دیگر کاربران تطبیق می دهد. اگر username تکراری باشد و یا password با تکرارش تفاوت داشته باشد، به کاربر ارور داده می شود و ورودی entry ای که اشتباه پر شده بود، پاک می شود.. در غیر اینصورت، اطلاعات کاربر جدید به User_Info.db اضافه می شود و همچنین اطلاعات کاربران و رمز ادمین بار دیگر از database دریافت می شود. در پایان، اگر ورود از طریق صفحهٔ start صورت گرفته باشد، کاربر به صفحهٔ کاربری جدیدی که ایجاد شده است، ارجاع داده می شود و اگر کاربر جدید توسط ادمین ایجاد شده باشد، تابع admin اجرا می شود.

5) آخرین تابع، تابع play است که سه پارامتر از جمله اطلاعات دو کاربر و تعداد سطرهای مورد نظر را دریافت می کند. در ابتدای تابع دو متغیر turn و filledCells به صورت global تعریف شده اند که turn نشان دهندهٔ بازیکنی است که نوبت حرکت با اوست و در ابتدای تعریف به صورت رندم انتخاب می شود و filledCells تعداد خانه هایی از جدول n*n است که پر شده اند. این تابع در ابتدای شروع بازی با دسترسی به اطلاعات دو بازیکن در database، تعداد بازی های آنها را یکی بیشتر می کند. سپس لیستی دوتایی تحت عنوان scores تعریف می شود که امتیازات دو بازیکن را نگه می دارد. صفحهٔ بازی که w نام دارد و در play ساخته می شود، دارای دو label است که در هنگام تعریف متن ندارند و متن و رنگ آنها با توابع showLabel1 و showLabel2 تعیین می شود. تابع showLabel1 در هر فراخوانی امتیازات بازیکنان را وارد label1 می کند و showLabel2 نوبت بازیکنان را مشخص می کند و با توجه به نوبت، رنگ label2 را نیز تغییر می دهد.

در زیر آنها، یک جدول n*n تشکیل می شود که هر خانه از این جدول با استفاده از کلاس cell طراحی شده است و این اشیا در لیستی به نام cells قابل دسترسی هستند. هر شی از کلاس cell، دارای x و y می باشد (که موقعیّت مکانی آنها در جدول و لیست cells را نشان می دهد) و در ظاهر، یک frame است که داخل آن یک label و دو button، یکی با عنوان S و دیگری O قرار دارند و به ترتیب متدهای fS و fO را فعال می کنند. در ابتدا label خالی از متن است ولی با فعال شدن fS یا fO، دو button با استفاده از تابع grid_remove ناپدید می شوند و متن و رنگ label به ترتیب با توجّه به حرف انتخاب شده توسط بازیکن و بازیکن انتخاب کننده انتخاب می شود. همچین این دو متد تابع added را فراخوانی می کند.

در ابتدا باید تابع formsSOS را بررسی کنیم. این تابع موقعیت مکانی یک cell و محتوای آن (O یا S) را دریافت می کند و تمامی حالاتی که ممکن است منجر به ایجاد SOS شود را بررسی کرده و در هر موردی که این بررسی موفّقیّت آمیز باشد، لیستی از سه خانه ای که در این SOS قرار دارند را به لیستی تحت عنوان listNeedToBeRed اضافه می کند و در پایان این لیست return می شود. در تابع added که در fS و fO فراخوانی می شود، ابتدا تعداد خانه های پرشده (filledCells) یکی اضافه می شود و سپس تابع

formsSOS فراخوانی شده و لیست برگردانده شده در l ریخته می شود. تعداد اعضای l نشان دهندۀ SOSهایی است که بازیکن در این حرکت موفق به تکمیلشان شده است؛ در نتیجه، این مقدار به امتیازات بازیکن اضافه می شود و سپس با تابع showLabel1، نمایش امتیازات بروزرسانی می شود. در صورتی که هیچ امتیازی توسط بازیکن کسب نشده باشد، نوبت عوض می شود و از طریق showLabel2 این تغییر به اطّلاع بازیکنان می رسد. در غیر اینصورت، تمامی خانه هایی که در لیست l قرار دارند با متد makeRed به رنگ قرمز در می آیند. در پایان اگر تمامی خانه ها پر شده باشند، تابع finishedGame اجرا می شود. در این تابع، در صورتی که امتیازات نابرابر باشند، برنده را از طریق messagebox به بازیکنان معرفی کرده و سپس در database به تعداد بازی های برده شدۀ بازیکن برنده، یکی اضافه می شود. در صورتی که امتیازات برابر باشند، تنها پیامی مبنی بر تساوی نشان داده می شود. در آخر، دوباره کاربر به صفحۀ start برگردانده می شود.

در زیر جدول بازی، دو button هر یک به رنگ بازیکن مورد نظر می باشند که با اجرای تابع guidance0 یا guidance1 به بازیکنان راهنمایی می کند. در این توابع، از تابع findSOS استفاده می شود که این تابع به ترتیب از تمامی cell ها می گذرد و با استفاده از تابع formsSOS بررسی می کند که در صورت قرار دادن S یا O آیا این حرکت منجر به تشکیل SOS می شود یا نه. در صورتی که پاسخ مثبت باشد، راهنمایی از طریق messagebox به اطلاع بازیکنان می رسد و عبارت True برگردانده می شود و در صورت منفی بودن پاسخ، برنامه گذشتن از cell ها را ادامه می دهد. اگر تا انتهای جدول هیچ پاسخ مثبتی در یافت نشد، پیامی مبنی بر عدم امکان کمک نشان داده می شود و False برگردانده می شود. در توابع guidance0 و guidance1، اگر نوبت بازیکن نباشد یا بازیکن هر سه راهنمایی خود را استفاده کرده باشد، سیستم ارور می دهد ولی در غیر اینصور، اگر نتیجه جستجو در cell ها موفّقیّت آمیز بوده باشد و True برگردانده شده باشد، تعداد کمک هایی که در متن button نشان داده شده بود یکی کمتر می شود. در آخر صفحۀ w، Exit button وجود دارد که با تابع exitP، در صورت اطمینان بازیکنان، از w خارج شده و dashboard اجرا می شود.

Github: https://github.com/TakeenJazayeri/SOS