Automation of Test Evolution

Student(s)

Barış Kerem Kurt

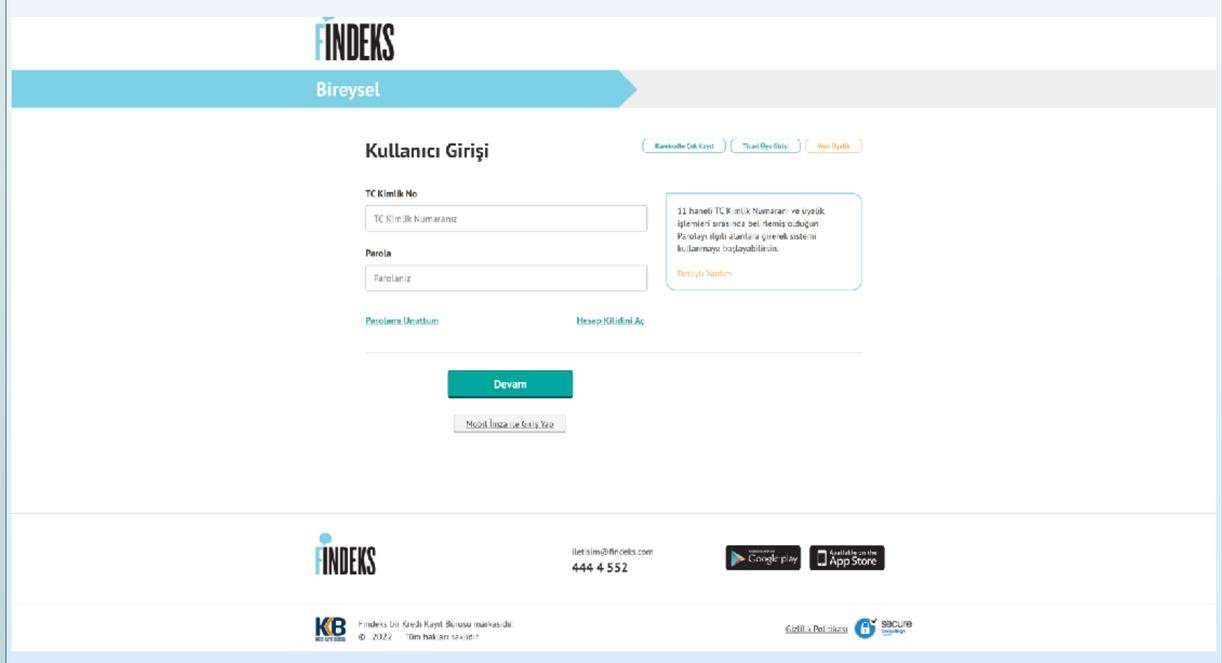
Faculty Member(s)

Cemal Yılmaz
Anıl Koyuncu

Company Advisor(s)

Melek Özlem Gökdeniz
Burak Çelik
Enes Akın
Ceren Kaplan
Hakan Yaman

ABSTRACT

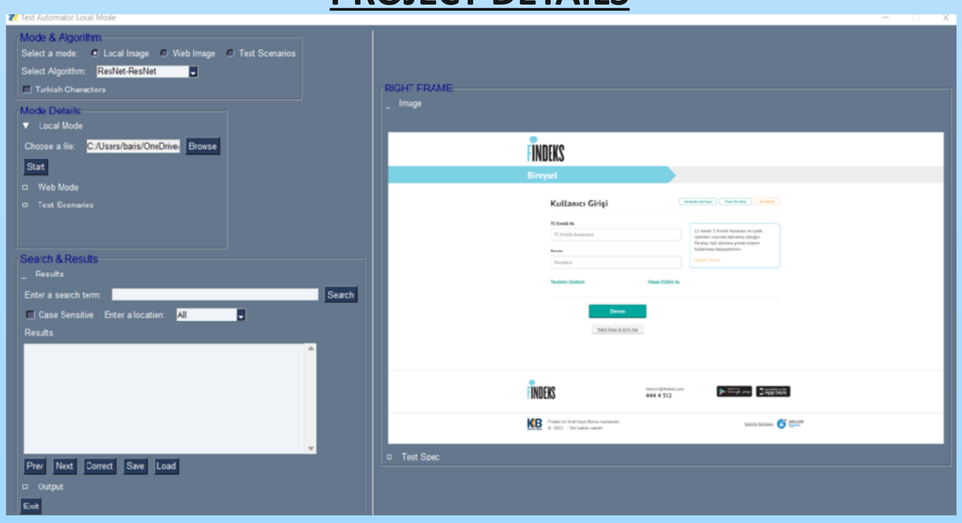


Automation of test evolution project is an industry-oriented project that carried out together with Kredi Kayıt Bürosu (KKB). KKB has a finance application called Findeks that is both available at web and mobile. Findeks application needs to be tested in order to improve the application quality and verify that application is working as designed. Findeks application consists of complex structures and many different pages. Therefore, a separate test case is required for each scenario to verify that the application is working correctly. Test cases often need an update. Constant breaking of test cases results in time and money loss at firm. That's why KKB needs an automated test tool for Findeks application which goes through each test scenario sequentially and automatically. In order to make an automated test tool, some integrations of software and algorithms will be done. Testing will be done via image processing, which is a sub-branch of machine learning. Basically, an image processing algorithm can locate UI elements and convert them into characters. In this way, automation is possible. Also, accuracy score is important to see whether the image processing algorithm did read the given image correctly. Another goal of the project is to compare different image processing algorithms and provide the best accuracy score. After the integration of different algorithms and software, automated test tool will be presented to the customer with a graphical user interface. Graphical user interface will make the tool easy to understand and user friendly.

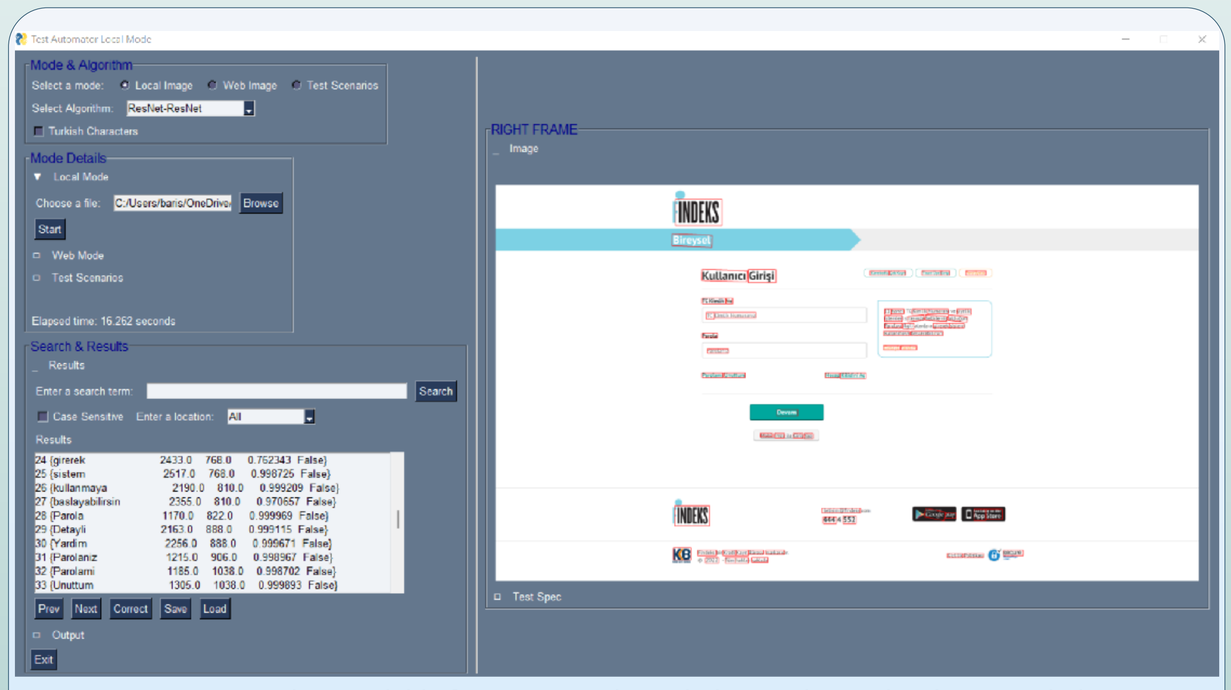
OBJECTIVES

- Learning and applying OCR technologies (ResNet, Tesseract, easyOCR)
- Modifying & mixing OCR Technologies to have higher accuracy results
- Learning and implementing end to end UI testing via BDD and Selenium
- Learning to manipulate Document object model (DOM) inside web browsers
- Gathering all objectives into a GUI which is the final product

PROJECT DETAILS



In GUI, user can select two modes for opening an image. Firstly, local mode is available for processing the images stored at local computer. If an image is selected, it will be displayed at the GUI. Secondly, web mode is available to process an image screenshot taken from an active web browser. Different from local mode, web mode has two buttons. one for launching the browser and another for taking screenshot and processing the image. User can also select OCR algorithms before starting the process. There are five different OCR algorithms as follows: Resnet, Tesseract, easyOCR, Resnet-Tesseract and easyOCR-Tesseract. Each algorithm have a different structure which results in different accuracy scores in different cases. OCR separates into text detection and text recognition. Text detection is responsible for finding the text locations inside the given image. Text recognition is responsible for interpreting the image texts to written text. Found texts have an accuracy score for determining the correctness of text. First three algorithms Resnet, Tesseract and easyOCR does the text detection and text recognition sequentially. On the other hand, 4th algorithm firstly does text detection via Resnet, then Tesseract does the text recognition. Similarly, 5th algorithm firstly does text detection via easyOCR, then Tesseract does the text recognition. After selecting the algorithm and image, user can start the process. Process may take several seconds for one image. After some processing time, results are displayed at GUI.



Test Scenarios tab is available for BDD. To achieve behavior driven development, behave framework was used at initial phase. In test scenarios, coordinates of the clickable element were found by Resnet OCR. Then found coordinates were given as input to the Selenium web automation browser. This allowed the selenium web automation browser to click and navigate between Findeks's pages. Selenium plays a big role because all test scenarios were built to run inside web browser. Furthermore, all test cases must be automatically done. Migrating to Gauge framework from Behave took some time. Because it was necessary to read and learn the Gauge documentation carefully. Ensuring a successfully working BDD framework allowed to make multiple Findeks test scenarios. With Findeks login scenario, the program can login to the Findeks application via finding the necessary elements using OCR, then clicking or editing the found element to continue. OCR search includes the necessary texts taken from the login page such as Turkish identity number input field, password input field and continue button. When OCR acquires the coordinates of these texts, the program continues firstly through clicking the Turkish identity number input field, then filling it. Similarly, password field will be found via OCR and clicked and filled via Selenium. Findeks will navigate to the next page once the button is clicked. The program is indefectible even if there will be an any change in the UI. Findeks login scenario requires several pages to navigate. After navigation and entering credentials, login scenario will be passed. In order not to find same results twice, position parameter is implemented. Furthermore, another step function is written to find elements in order. This function is useful when there are similar clickable elements on the screen. Position information allowed program to navigate faultless even there are more than one same text found on the specific screen. After handling basic navigation successfully, the next challenge was finding specific elements and differentiate them. To separate elements, web browser's document object model stepped in. DOM helped for finding html tags. Moreover, page analyze function gets the text and coordinate information of all visible elements on the screen to achieve ground truth. DOM is further used to find elements that OCR cannot find. These items are input bars, dropdowns, and calendars. Resnet OCR can only find text bounding boxes. However, via DOM, program can find elements that does not include any text. A good example is filling an input bar which has no placeholder text. In most cases, every input bar has a label near to them. This is also visible inside the DOM hierarchy. Starting from label, siblings, or parents of label html tag, can be found via DOM hierarchy. Label's closest input html tag is most likely to be the corresponding input bar. That's why, function firstly gets the label from OCR. OCR returns a coordinate, then function finds the element on given coordinates which will be a label. Then from DOM, function searches the closest wanted elements. This can be input bar, dropdown, calendar, or other elements that OCR cannot find. To achieve this, DOM search is implemented via breadth-first search.

CONCLUSIONS

Through the Fall 2021 and Spring 2022 Terms, most of the objectives and tasks for Automation of Test Evolution project have been accomplished. Related methodologies were learnt and applied onto project. As a result, project has a complete working image processing algorithms integrated with a graphical user interface. In addition to GUI and image processing algorithms, an automated testing tool Selenium is used for Test Automator Web Mode for opening web browsers. Also, behavior driven development (BDD) test cases were written. Also, BDD test cases are integrated with the project's main software. Program can navigate through Findeks web application's pages using optical character recognition and Selenium web driver. Real time pictures of browser are given as input to ResNet OCR at each page change. Then, necessary coordinates are taken from OCR output and given to Selenium web browser. Selenium web browser does the interaction operations such as filling an input bar or clicking to a button. Various test cases written at a very close language to KKB's used BDD definitions.

REFERENCES

- He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2016). Deep Residual Learning for Image Recognition.
- Martinez, Dennis. (2021). Top Challenges of Automated End-to-End Testing.
- Rehkopf, Max. (2021). Automated software testing.
- Rosebrock, Adrian. (2020). Getting started with EasyOCR for Optical Character Recognition.
- Smith, Ray. (2007). An Overview of the Tesseract OCR Engine.