11 System Manual

11.1 Frontend

The client-side of web application is implemented with using ReactJS library, on Javascript. In order to run client-side application, you need to install node first. Then you can run these commands below:

- 'npm install' to install project dependencies.
- 'npm start' to run the app in the development mode. Open $http://localhost:3000$ to view it in the browser. The page will reload if you make edits. You will also see any lint errors in the console.
- 'npm run builds' Builds the app for production to the 'build' folder. It correctly bundles React in production mode and optimizes the build for the best performance. The build is minified and the filenames include the hashes. The app is ready to be deployed. Then, serve the static contents using serve by running these commands npm install -g serve and then serve -s build -l <PORT>
- npm run lint to lint code
- npm run lint – –fix to auto fix linter errors.

Browsers Support We test our application on

- Chrome 79.0.3945.88
- Firefox 71.0

11.2 Android

11.2.1 Software Description

Android software is simply an android application of Khaji-it project and provides simple user interfaces for clients and applies all the features of the Khaji-it API offers.

11.2.2 Benefits Value

The benefits of using with android software, developers may:

- Use API interface developed within Android environment
- Implement new use case scenarios based on provided layouts
- Implement new use case scenarios based on provided layouts

11.2.3 Platform Requirements

Requirements for android software can be listed as:

- Android SDK v29 (minSdkVersion: 24, targetSdkVersion: 29)
- Kotlin Android
- Dependencies: ,
  - Android Support Libraries (activity:1.0.0, annotation:1.1.0,appcompat-resources:1.1.0,appcompat:1.1.0, core-common:2.1.0, core-runtime:2.1.0, asynclayoutinflater:1.0.0,cardview:1.0.0,collection:1.1.0, constraintlayout-solver:1.1.3, constraintlayout:1.1.3, coordinatorlayout:1.1.0-beta01, core-ktx:1.1.0, core:1.1.0,cursoradapter:1.0.0, customview:1.0.0, databinding-adapters:3.5.1, junit:4.12 ,viewpager:1.0.0, versionedparcelable:1.1.0, espresso-core:3.2.0, recyclerview:1.1.0-beta04, databinding-common:3.5.1, fragment:1.1.0, lifecycle-common:2.1.0)
  - Square Libraries (okio:1.15.0, picasso:2.71828, retrofit:2.5.0, gson:2.8.2, viewpager2:1.0.0-beta04)
  - Other Libraries (kxml2:2.3.0, httpclient:4.0.1, httpcore:4.0.1, hamcrest-core:1.3, hamcrest-integration:1.3, hamcrest-library:1.3, kotlin-android-extensions-runtime:1.3.50, kotlin-stdlib-common:1.3.50, kotlin-stdlib-jdk7:1.3.50, kotlin-stdlib:1.3.50, kotlinx-coroutines-android:1.3.2, kotlinx-coroutines-core:1.3.2, glide:4.4.0,result:1.6.0, autoimages-lider:1.3.2, auth-api-impl:11.6.0, play-services-auth-api-phone:17.0.0, play-services-auth-base:17.0.0, play-services-auth:17.0.0, play-services-base:17.0.0, play-services-basement:17.0.0, play-services-location:17.0.0, play-services-places-placereport:17.0.0, play-services-tasks:17.0.0)

Khaji-it android software was developed in Kotlin language with Retrofit library which turns the project's HTTP API into a Java interface. In addition to Retrofit, Glide library is used to display images with their Uri provided from Khaji-it API. Besides, one may use Java as a language and take advantage of Java files which are generated from implemented Kotlin software. You may want to use Android Studio as development environment since project was developed within Android Studio and below descriptions include some usage of it.

11.2.4 Importing Project to Android Studio

File → New → Project from Version Control → Git
URL: https://github.com/bounswe/bounswe2019group1 After importing project, you will see Android project files within Android scope in the Project panel. You can view and edit Gradle Scripts. There are two Gradle Scripts: one for app and one for Android. We suggest that editing Gradle Script for app is enough. Project file structure can be seen in app package. Implemented package name is "android.khajit".
If you want to see or edit implemented Kotlin software, the following path will take you to the required folder: app → src → main → java → com.project.khajit_app.activity . There are packages (folders) and MainActivity file.

11.2.5 Software Details

Interfaces

- interfaces.IOnBackPressed
- interfaces.fragmentOperationsInterface.kt

Activity

- com.project.khajit_app.activity.HomeFeedPageActivity:
- com.project.khajit_app.activity.HomeFeedPageActivity:
- com.project.khajit_app.activity.Globals:
  Implements global variables during a session of a user.
- com.project.khajit_app.activity.LoginPageActivity:
  Implements login action of a user.

- com.project.khajit_app.activity.HomeFeedPageGuestActivity:
- com.project.khajit_app.activity.UserViewAdapters:
- com.project.khajit_app.activity.SignUpPageTraderActivity:
- com.project.khajit_app.activity.FollowListViewAdapter:
- com.project.khajit_app.activity.OtherPortfolioListAdapter:
- com.project.khajit_app.activity.PortfolioListAdapter:
- com.project.khajit_app.activity.HelperFunctions:
- com.project.khajit_app.activity.OnePortfolioViewAdapter:
- com.project.khajit_app.activity.ui:
  - com.project.khajit_app.activity.ui.annotation:
  - com.project.khajit_app.activity.ui.article:
  - com.project.khajit_app.activity.ui.editprofile:
  - com.project.khajit_app.activity.ui.event:
  - com.project.khajit_app.activity.ui.followlist:
  - com.project.khajit_app.activity.ui.home:
  - com.project.khajit_app.activity.ui.myportfolio:
  - com.project.khajit_app.activity.ui.mywallet:
  - com.project.khajit_app.activity.ui.notification:
  - com.project.khajit_app.activity.ui.otherprofile:
  - com.project.khajit_app.activity.ui.otherportfolio:
  - com.project.khajit_app.activity.ui.prediction:
  - com.project.khajit_app.activity.ui.search:
  - com.project.khajit_app.activity.ui.equipment:
  - com.project.khajit_app.activity.ui.profile:

API

- com.project.khajit_app.api.Api:

- com.project.khajit_app.api.AnnotationApi:
- com.project.khajit_app.api.RetrofitClient:

Data

- com.project.khajit_app.data.annotationModels:
  - com.project.khajit_app.data.annotationModels.AddBodyModel:
  - com.project.khajit_app.data.annotationModels.AnnotationModelResponse:
  - com.project.khajit_app.data.annotationModels.BodyModel:
  - com.project.khajit_app.data.annotationModels.CreateAnnotationModel:
  - com.project.khajit_app.data.annotationModels.CreateAnnotationModelNewCreator:
  - com.project.khajit_app.data.annotationModels.CreateAnnotationResponse:
  - com.project.khajit_app.data.annotationModels.CreatorExistsResponse:
  - com.project.khajit_app.data.annotationModels.CreatorListModel:
  - com.project.khajit_app.data.annotationModels.CreatorModel:
  - com.project.khajit_app.data.annotationModels.DeleteAnnotationModel:
  - com.project.khajit_app.data.annotationModels.DeleteAnnotationResponse:
  - com.project.khajit_app.data.annotationModels.GenericAnnotationModel:
  - com.project.khajit_app.data.annotationModels.GetAnnotationModelResponse:
  - com.project.khajit_app.data.annotationModels.RefinedByModel:
  - com.project.khajit_app.data.annotationModels.SelectorModel:
  - com.project.khajit_app.data.annotationModels.ShowImageAnnotationModel:
  - com.project.khajit_app.data.annotationModels.ShowTextAnnotationModel:
  - com.project.khajit_app.data.annotationModels.TargetModel:
  - com.project.khajit_app.data.annotationModels.sourceModel:
- com.project.khajit_app.data.Models:
  - com.project.khajit_app.data.modelsArticleCommentItem:
  - com.project.khajit_app.data.modelsArticleDislikeResponseModel:
  - com.project.khajit_app.data.modelsArticleLikeDisLikeResponseModel:

- com.project.khajit_app.data.modelsArticleLikeDislikeCountResponseModel:
- com.project.khajit_app.data.modelsArticleLikeDislikeModel:
- com.project.khajit_app.data.modelsArticleLikeResponseModel:
- com.project.khajit_app.data.modelsArticleSearchModelResponse:
- com.project.khajit_app.data.modelsArticleSearchResponse:
- com.project.khajit_app.data.modelsBasicRegisterResponse:
- com.project.khajit_app.data.modelsBasicUser:
- com.project.khajit_app.data.modelsChangePrivacy:
- com.project.khajit_app.data.modelsCommodityResponse:
- com.project.khajit_app.data.modelsCreateArticleModel:
- com.project.khajit_app.data.modelsCreateArticleResponseModel:
- com.project.khajit_app.data.modelsCreateCommentModel:
- com.project.khajit_app.data.modelsCreateCommentResponseModel:
- com.project.khajit_app.data.modelsCryptoResponse:
- com.project.khajit_app.data.modelsCurrencyResponse:
- com.project.khajit_app.data.modelsDepositFundsModel:
- com.project.khajit_app.data.modelsDepositFundsResponse:
- com.project.khajit_app.data.modelsETFResponse:
- com.project.khajit_app.data.modelsEquipmentBSModel:
- com.project.khajit_app.data.modelsEquipmentBSOrderModel:
- com.project.khajit_app.data.modelsEquipmentBSOrderResponse:
- com.project.khajit_app.data.modelsEventModel:
- com.project.khajit_app.data.modelsFollowIDModel:
- com.project.khajit_app.data.modelsFollowIDModelResponse:
- com.project.khajit_app.data.modelsFollowIdListRequestModel:
- com.project.khajit_app.data.modelsFollowModel:
- com.project.khajit_app.data.modelsFollowModel2:
- com.project.khajit_app.data.modelsFollowUnfollowModel:
- com.project.khajit_app.data.modelsFollowUnfollowResponseModel:

- com.project.khajit_app.data.modelsFollowingPendingListResponseModel:
- com.project.khajit_app.data.modelsFollowingPendingResponseModel:
- com.project.khajit_app.data.modelsGeneralArticleModel:
- com.project.khajit_app.data.modelsGeneralArticleSpannableModel:
- com.project.khajit_app.data.modelsGeneralFollowModel:
- com.project.khajit_app.data.modelsGeneralFollowModel2:
- com.project.khajit_app.data.modelsGeneralNotificationModel:
- com.project.khajit_app.data.modelsGenericLoginModel:
- com.project.khajit_app.data.modelsGenericUserModel:
- com.project.khajit_app.data.modelsListArticleCommentModel:
- com.project.khajit_app.data.modelsListEventModel:
- com.project.khajit_app.data.modelsListEventResponse:
- com.project.khajit_app.data.modelsListNotificationsResponse:
- com.project.khajit_app.data.modelsListPortfolioResponse:
- com.project.khajit_app.data.modelsLoggedInUser:
- com.project.khajit_app.data.modelsLoginResponse:
- com.project.khajit_app.data.modelsOneEventResponse:
- com.project.khajit_app.data.modelsOnePortfolioResponse:
- com.project.khajit_app.data.modelsPasswordChange:
- com.project.khajit_app.data.modelsPendingFollowerResponse:
- com.project.khajit_app.data.modelsPortfolioDeleteResponseModel:
- com.project.khajit_app.data.modelsPortfolioEditRequestModel:
- com.project.khajit_app.data.modelsPortfolioEditResponseModel:
- com.project.khajit_app.data.modelsPortfolioModel:
- com.project.khajit_app.data.modelsPortfolioOwnerModel:
- com.project.khajit_app.data.modelsPredictionModel:
- com.project.khajit_app.data.modelsPredictionResponseModel:
- com.project.khajit_app.data.modelsPublicArticleListResponse:
- com.project.khajit_app.data.modelsSearchRequest:

- com.project.khajit_app.data.modelsSearchResponse:
- com.project.khajit_app.data.modelsShowArticleCommentModel:
- com.project.khajit_app.data.modelsStockResponse:
- com.project.khajit_app.data.modelsTradeIndiceResponse:
- com.project.khajit_app.data.modelsTraderUser:
- com.project.khajit_app.data.modelsUpdateUser:
- com.project.khajit_app.data.modelsUpdateUserResponse:
- com.project.khajit_app.data.modelsUpgradeDowngrade:
- com.project.khajit_app.data.modelsUserAllInfo:
- com.project.khajit_app.data.modelsUserInfoGet:
- com.project.khajit_app.data.modelsWalletResponse:
- com.project.khajit_app.data.modelscreateWalletResponse:
- com.project.khajit_app.data.modelsisFollowingPortfolioResponse:
- com.project.khajit_app.data.modelsisFollowingResponseModel:
- com.project.khajit_app.data.modelsuserToBeLogin:

Global

- com.project.khajit_app.global.User:

Service

- com.project.khajit_app.service.FetchAddressIntentService

11.3 Backend

We developed backend of this project with Python 3.7 and Django 2.2.6. To provide requlority and compatibility we create requirements.txt and use pip freeze to write libraries we used. In this point guide for backend divides two part: To run the code in local with sqlite and to run the code with docker ıver postgresql. Since we use docker to run in frontend and mobile team we try to maximize usability of docker. So we add every command to setup our application in Dockerfile and docker-compose.yml. Running code in local with sqlite mostly used in backend team because set up is more complicated but development with this procedure is easier to use.

11.3.1 Set Up In Local

Before running this project first you have to install every library we are using. If you want to user virtual environment:
To create virtual environment the command should be :

- virtualenv "name_of_virtualenv"

then to activate:
for windows:

- source name_of_virtualenv/bin/activate

for mac:

- . name_of_virtualenv/bin/activate

You should create virtual environment for only first time but activate every time.

Then you should install packages in requirements if any new package was added to project :

- pip install -r requirements.txt

Then the project is ready to go. To create or update database:

- python manage.py makemigrations –settings=tradersplatform.settings.local

- python manage.py migrate –settings=tradersplatform.settings.local

After that you can run the project:

- python manage.py runserver –settings=traderspl atform.settings.local

*Base settings were created according to the docker file so user who tries to run with python manage.py should also give the settings.
*Also if both python2 and python3 are installed in your machine or virtualenvironment you should specify the version of python in every python command like:

- python3 manage.py runserver –settings=traderspl atform.settings.local

11.3.2 Set Up With Docker

Witth docker setting up and running is much more easier because every database commands and run commands exists in dockerfile and docker-compose.yml. Only steps you should do will be :

- Installing docker
- Go the directory of docker-compose.yml file
- Run docker-compose build
- Run docker-compose up
- If any change in database was made docker-compose down then docker-compose up
- If any change in libraries was made docker-compose down then docker-compose build docker-compose up

11.4 Annotation Backend

Before running this project first you have to install every library we are using. If you want to user virtual environment:
To create virtual environment the command should be :

- virtualenv "name_of_virtualenv"

then to activate:
for windows:

- source name_of_virtualenv/bin/activate

for mac:

- . name_of_virtualenv/bin/activate

You should create virtual environment for only first time but activate every time.

Then you should install packages in requirements if any new package was added to project :

- pip install -r requirements.txt

Then the project is ready to go. To create or update database:

- python manage.py makemigrations

- python manage.py migrate

After that you can run the project:

- python manage.py runserver 0.0.0.0:8020

*Annotation should be served in 8020 port according to our design *Also if both python2 and python3 are installed in your machine or virtualenvironment you should specify the version of python in every python command like:

- python3 manage.py runserver 0.0.0.0:8020