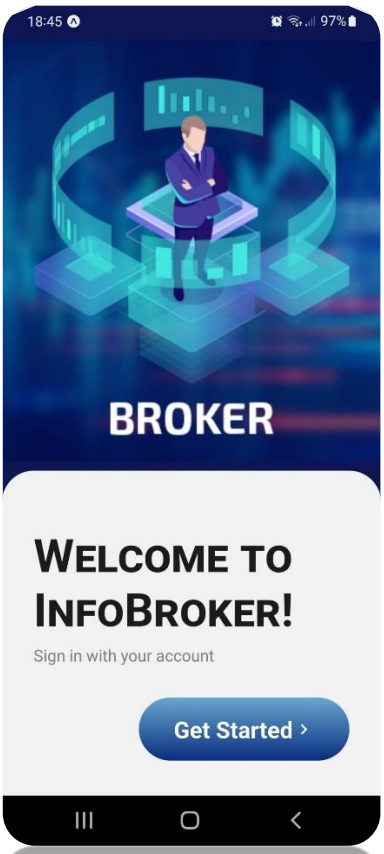
Moshe Crespin 315344515
Rotem Halbreich 311549364

Middle Submission:

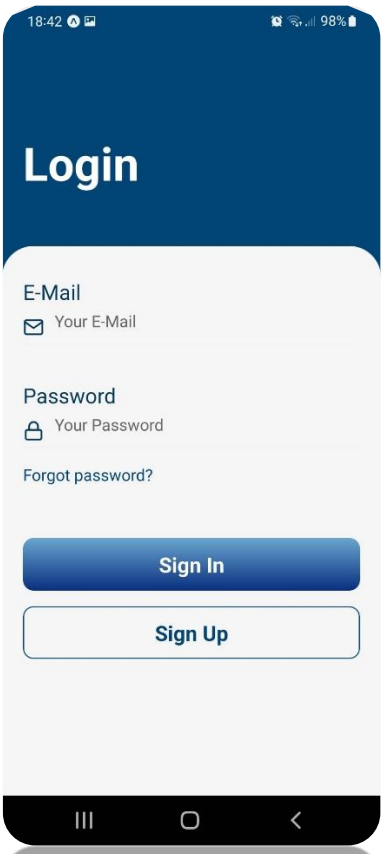
GitHub Front-End: https://github.com/RotemHalbreich/InfoBroker.git

GitHub Back-End: https://github.com/mosheCrespin/infoBroker_backEnd.git

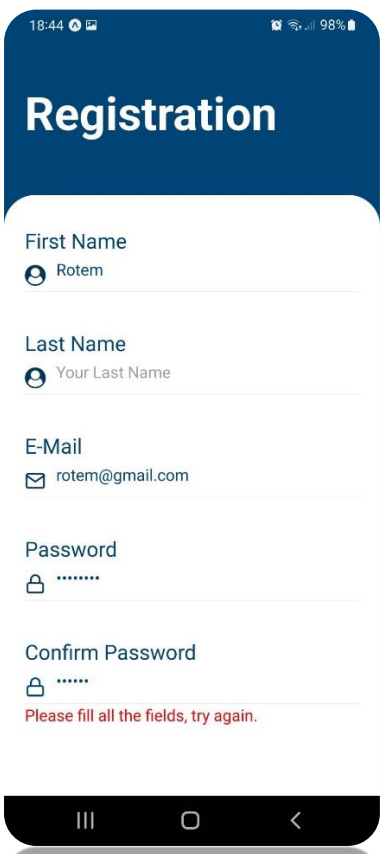
Splash opening screen



Login screen

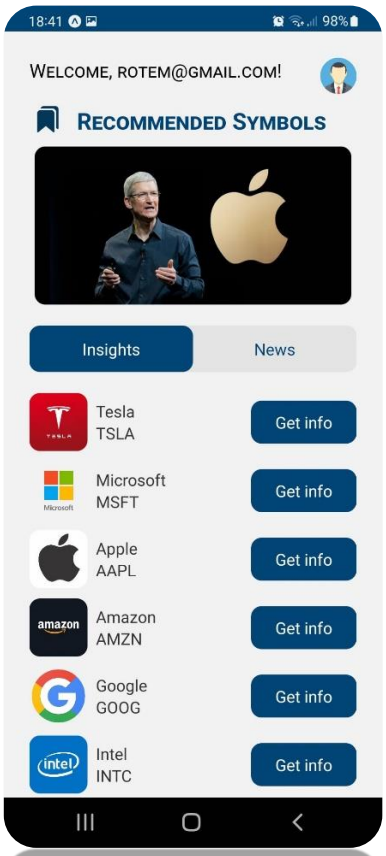


Registration screen

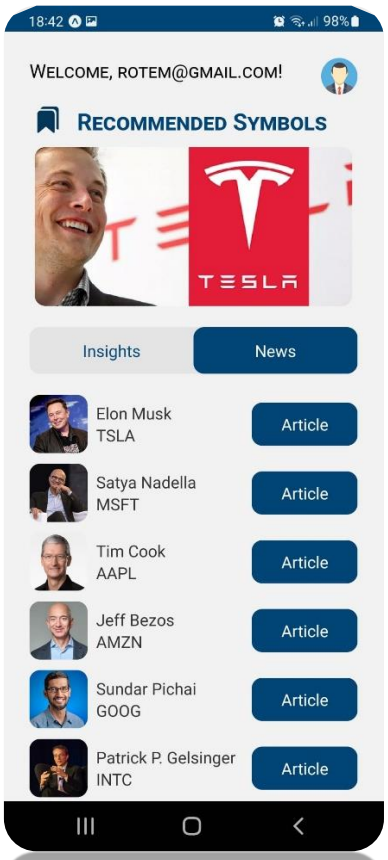


Moshe Crespin 315344515
Rotem Halbreich 311549364

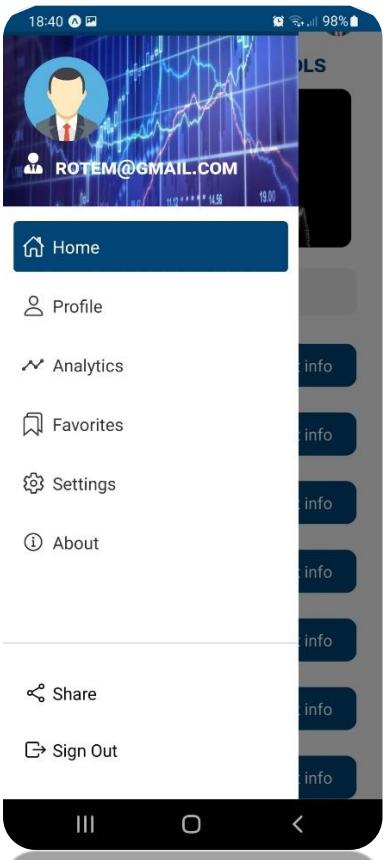
Home screen Insights



Home screen News

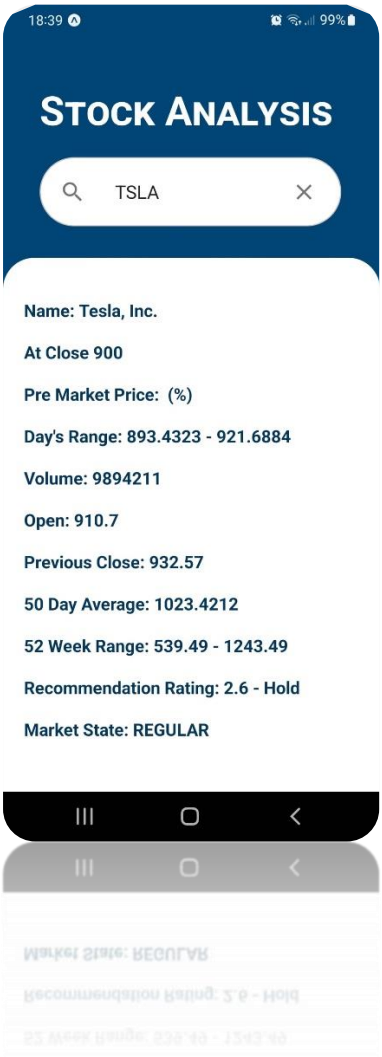


Sidebar



Moshe Crespin 315344515
Rotem Halbreich 311549364

Stock Analysis



Moshe Crespin 315344515
Rotem Halbreich 311549364

Objects & Components:

Front-End:

App:

- The Application's skeleton which unites and holds all screens in a specific order using Navigation Container to navigate between them.

Screens:

- **SplashScreen.js** – This is our opening screen presenting the App's Logo and welcoming the new and returning users.
- **SignInScreen.js –** Check validation of email, confirmed password, then it sends the given data to the server. If the server sends a token, then we save the token and navigate the user to the home page. O.W we display an error massage to the user.
- **SignUpScreen.js –** Checks validation of email, confirms password, and if the user has filled all the fields, if so, then sends it to the server. If the server sends a token as a response, then we store the token and navigate the user to the home page. Otherwise, we display an error message to the user.
- **resolveAuthScreen.js –** When the user enters to our App this screen wakes up, this screen checks if there is already a token stored in the user device, if so, we navigate the user to the home screen, O.W we navigate the user to the sign in screen.
- **AnalyzeScreen.js –** Searching a specific stock using a search bar, sending a request over to an API to get the relevant data from the server side.

Moshe Crespin 315344515
Rotem Halbreich 311549364

Components:

- **CustomDrawer.js –** A custom drawer's framework which is called in the App's Drawer Navigation Container, holding the user's avatar, username & bottom buttons for sharing the App & Signing out. Sign out – from the navigation bar: If the user wants to sign out, then we remove the stored token and navigate the user to the login page.
- **CustomSwitch.js –** A custom switch for the home screen which lets you move between 2 options (Insights/News), each of them holds our data.
- **BannerSlider.js** – A custom banner used in the home screen's carousel when rendering the items.
- **ListItem.js –** A custom list of items used in the home screen for the Insights & News lists.

Navigation:

- **StackNavigation.js –** This is the Stack navigation used to navigate between the authentication screens until you get to the home screen.

Back-End:

Db:

- **connect.js –** It's job is to connect to the fire base data base
- json with the key for the firebase API

Routes:

- **Stock.js -** Gives the initialize route for the stock routes
- **Auth.js -** Gives the initialize route for the stock routes

Models:

- **Stock.js -** Scheme for a stock

Moshe Crespin 315344515
Rotem Halbreich 311549364

- **User.js -** Scheme for a user

Middleware:

- **Authenticatio.js:** the purpose of this file is to check the given massage and process it. I will focus here about the functions in this file:
  - All_element_exists: in register, this function check if all the elements provided.
  - User_already_exists: in register, this function check if the given mail already exists in the DB, if so, it returns error.
  - User_not_exists: for login, if the given user is **not** exists then it returns error.
  - PasswordHash- in register- get the given password and hash it, we are not store the password in plain text.
  - VerifyToken: when a user register or login he receives a token, for any API request the client side send the token with this token (inside the header), this function checks if the given token is valid, if not, it returns an error.

Controller:

- **Auth.js:**
  - Register: gets the data from the message, saves the new user in the database, and sends a token as a response.
  - Login: get the data from the message, get the given user password from the DB, compare between the passwords, if match then returns a new token, if not returns an error.
- **Shares.js:**
  - GetCurrStockData: gets a symbol from the client, sends to a third-party API to get the current stock data, takes the relevant data, process it and sends the relevant data to the client as a response.