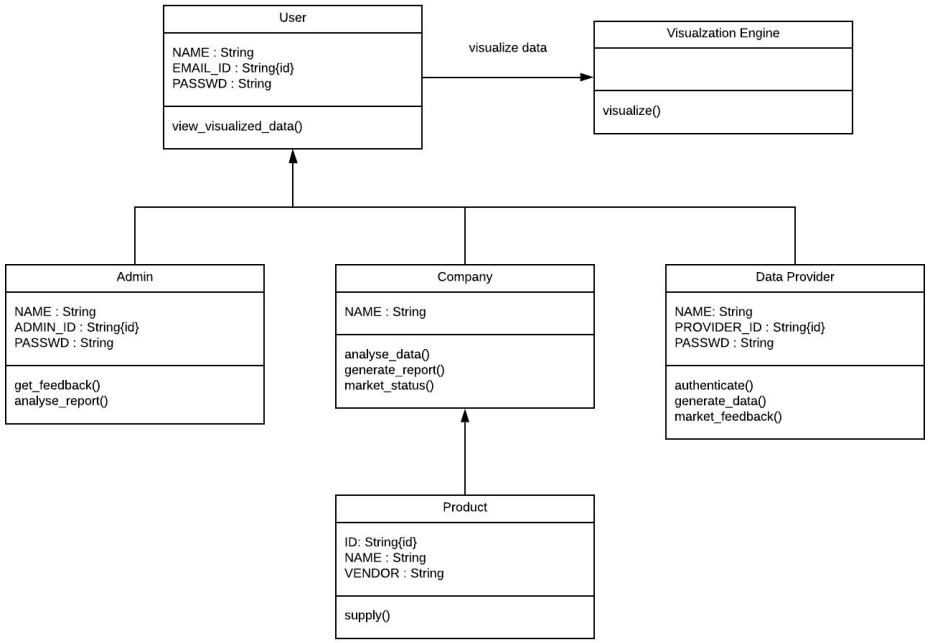
PROJECT ID : P07

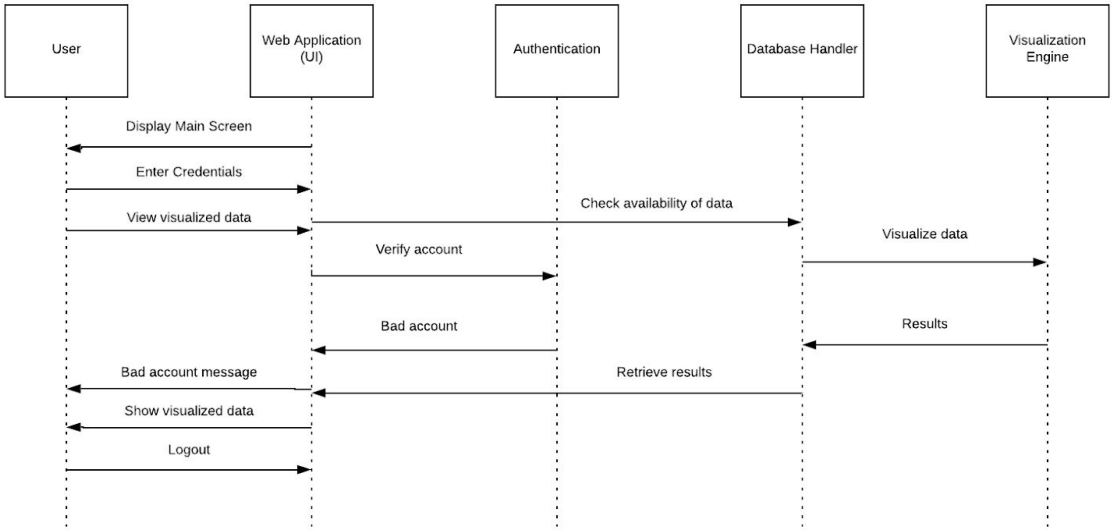
NAME : ATHARVA KOKATE **REGNO : 18BCE0709**

MODULE : { DATA_VISUALIZATION & REPRESENTATION }

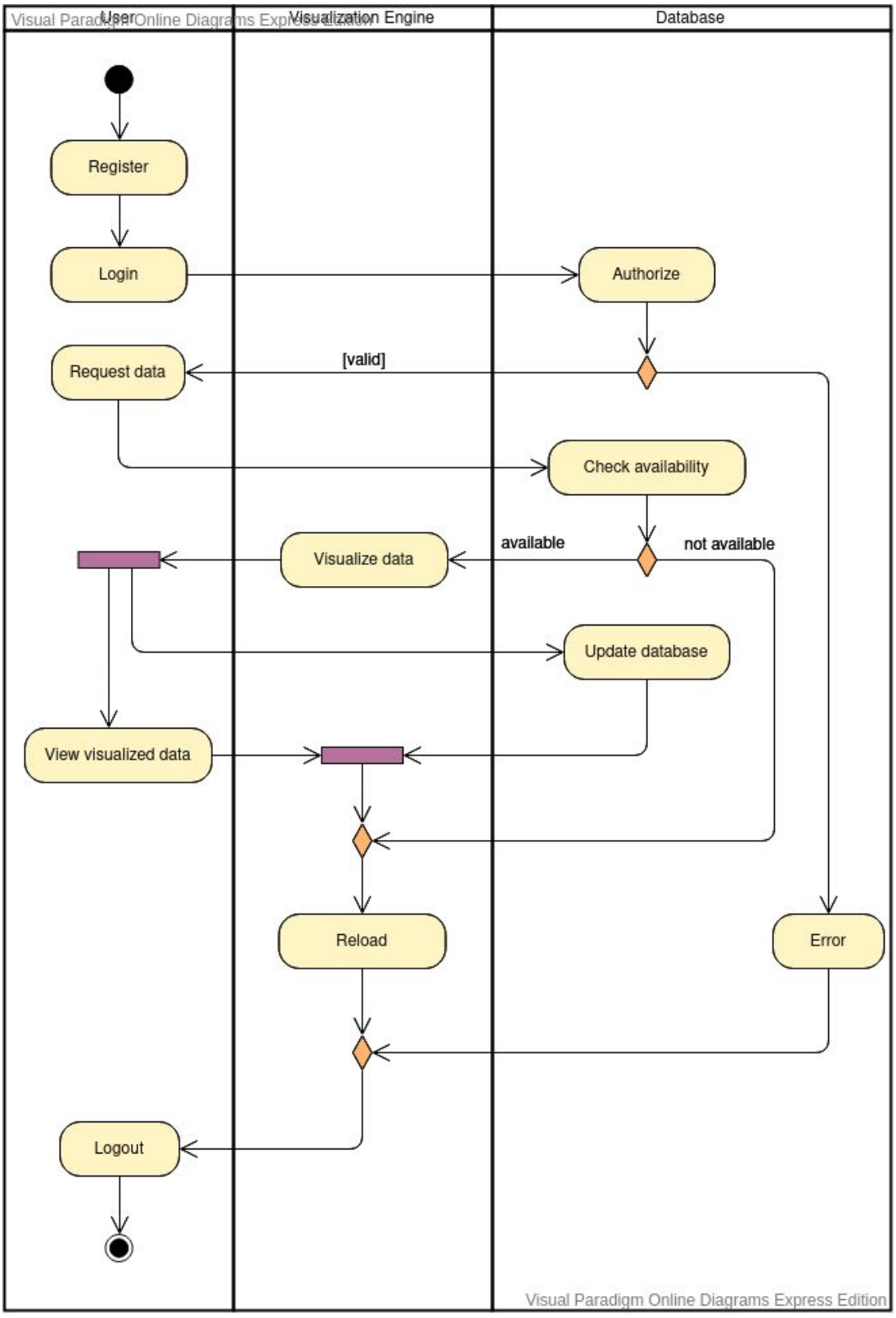
ABSTRACT LEVEL DIAGRAM (CLASS DIAGRAM)



SEQUENCE DIAGRAM



ACTIVITY DIAGRAM



EXPLANATION

The part of novel idea that I am handling is the VISUALIZATION module. Along with the visualization module I am also handling the stock prediction part of the project. The prediction part uses an artificial recurrent neural network called Long Short Term Memory (LSTM) to predict the closing price of a corporation (In this case the company using the software) using the past 60 day stock price. In the visualization module, I am using chart.js for visualization of graphs and quotes so that company can easily analyse all the factors affecting their market control and hence increase the probability of success for the product. Chart.js is a javascript module that helps in making interactive graphs. I will get the stock quote for the past few years from yahoo (probably past five years) and visualize the closing price history for the raw data. Then I will create a new dataframe with only the 'close' column and train the LSTM model after scaling the training dataset. After training, I will test the model with the testing dataset and plot the predicted price values with the matplotlib module. After plotting there are two ways - (1) I can export the image of the plot as png using matplotlib which can be used further with node to display in the software/website. This will work for sure in the short run but might have performance issues in the long run. So the other option is (2) to export the plot data to node and use it with the help of javascript and chart.js to visualize in the website. I am using a combination of both ways as because the starting data for the model to process is less and can work fine for visualizations in the website. And for the static graphs or for the graphs that are not very dynamic chart.js will be used.

The sequence diagram will give proper understanding of the model and shows the working flow and behaviour of the model.

USER

The user will interact with the web application user interface and request for the visualizations of the trained and tested models. If he is authenticated, further request is made to the database to check if there is any data available for the requested model.

WEB APPLICATION (UI)

The web application acts as an interface between the user and different models. It maps the inputs from the user to different functionalities of the trained models. Also it helps user to authenticate the user by connecting the user to the authentication service provider (in this case passport).

AUTHENTICATION

Here passport in node js will handle all the authentication services. The browser will send the request to the server express app (node) which will process it and send the request to the OAuth provider. The OAuth provider will process the request, verify the user, fetch his details and send it back to the server express app. It will receive the user details from the provider and lookup/ create user in its own database (I am using mongodb here). After that it will create a unique cookie and send it to the browser. Server Express App decodes cookie and retrieves user information. Browser stores the cookie until the session is active.

DATABASE HANDLER

I am using mongodb here. Mongodb is a no-sql database, easy to setup and handle. Mongodb will store user's information and the model's dataset. Once the request for visualization is made, it will send the relevant data to the visualization engine.

VISUALIZATION ENGINE

Here I am using chart.js for visualization. Once data comes from the database, the only task remains is to fit it into chart.js data format. After visualizing the data it will update the logs in the users profile and send the result to the web application UI to display it to the user. Then the user visualizes the data.

IN CASE OF ERROR OR BAD REQUEST

The user will be redirected to the start of the process in case of any error as processing a bad request can be harmful for the model. The error message will be displayed to him.
