YES Business Market Simulator Models

The aim of this document is to provide insight into the system models and their attributes.

Diagram

Description automatically generated

Contents

[User Model 3](#_Toc55414694)

[YES Model 3](#_Toc55414695)

[School Model 3](#_Toc55414696)

[Team Model 4](#_Toc55414697)

[Strategy Model 4](#_Toc55414698)

[StrategyPolicy Model 5](#_Toc55414699)

[PolicyType Model 5](#_Toc55414700)

[Policy Model 5](#_Toc55414701)

[MarketEntry Model 6](#_Toc55414702)

[MarketAttributeType Model 6](#_Toc55414703)

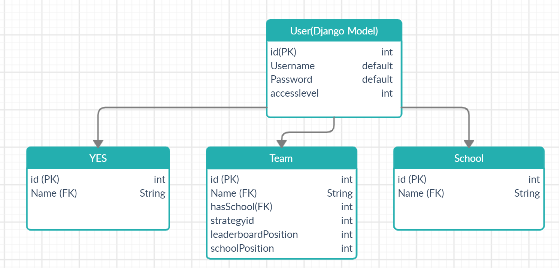
[MarketAttributeTypeData Model 7](#_Toc55414704)

[Simulator Model 8](#_Toc55414705)

[MarketEvent Model 9](#_Toc55414706)

[PolicyEvent Model 9](#_Toc55414707)

# User Model



The user model is native to Django model and provides the Username and password attributes.

accesslevel will be implemented using Django groups (More info: <https://medium.com/djangotube/django-roles-groups-and-permissions-introduction-a54d1070544> ) accesslevel will be used to differentiate the types of users and most importantly what content they can access. The three types of user groups will be ‘staff’, ‘school’, ‘team’.

## YES Model

This model extends the User model

The name attribute is the YES staff members first name, they should be able to change this at any time via their dashboard/account page

## School Model

This model extends the User model

The name attribute is the schools name, the school should be able to change this at any time via their dashboard/account page.

## Team Model

This model extends the User model

The **name** attribute is the teams name, the school should be able to change this at any time via their dashboard/account page, the team cannot change their own name that the school has assigned them. This is for child protection purposes.

The **hasSchool** attribute is a foreign key, one-to-one field, which points to the school id which the team belongs to. Storing the school that the team belongs to is necessary so their position in the school leader board can be calculated, and so that the school may edit the teams account if necessary.

The **strategyid** attribute is a foreign key, one-to-one field, which points to the strategy model which stores the data regarding the teams strategy.

The **leaderboardPosition** attribute is an integer field which stores the teams rank amongst other teams. The leaderboardPosition will initially be null and will be updated and recalculated at the end of every trading day once the sales have been distributed.

The **schoolPosition** attribute is an integer field which stores the teams rank amongst their school. Like leaderboardPosition, it will initially be null and then updated and recalculated at the end of every trading day once the sales have been distributed.

# Strategy Model



The **teamid** attribute is a foreign key, one-to-one field, pointing to the team the strategy instance represents.

The **consistency [not implemented]** attribute is a bonus multiplier awarded to teams who do not modify their strategy. The consistency multiplier will be taken into account when deciding the number of sales a team receives. At the end of each trading day the consistency attribute will increment by a hardcoded x. Editing the strategy should reset the consistency attribute to 0.

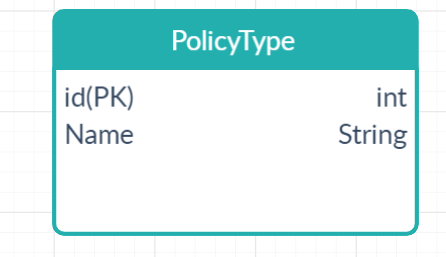
# StrategyPolicy Model



This model acts as a many-to-many relationship between strategy and policy table.

Django provides a many-to-many relationship attribute. This table may be implemented as a many-to-many field in each table (PolicyType and Strategy).

# PolicyType Model



The **name** attribute stores the label used to for the policy. From the requirements specifications, the identified PolicyType’s are ‘Strategy’, ‘Priorities and Management’, ‘Market Development’, ‘Product Inputs’. These labels can be changed by YES in between games if necessary.

# Policy Model



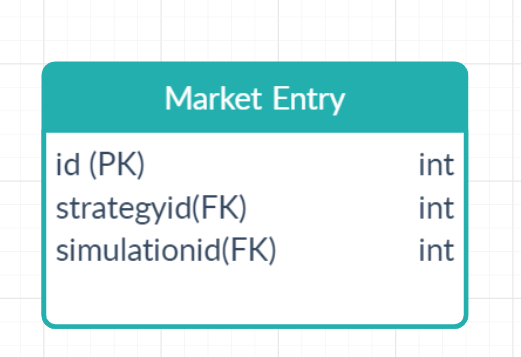
The **name** attribute stores the label used to identify the policy option. The policy options for a policy type ‘Strategy’ could include ‘Product Quality’, ‘Serving Speed’ or Staff Training‘. The YES staff should be able to change the labelling for this in between games.

The **weightingFactor** attribute is a real number which is fed into the sales allocation calculation. It would be nice for the YES members of staff to be able to edit these in-between games if they choose to do so.

The **belongsTo** is a foreign key, one-to-many (one PolicyType to many Policies), attribute which points to the PolicyType the policy belongs to.

# MarketEntry Model

This model is used to store the performance of a team’s product within a market. It also acts as a link between a simulation and the teams taking part.

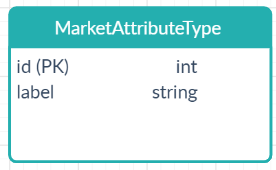


The **strategyid** is a one-to-one foreign key attribute pointing to the strategy profile for a given team.

The **simulationid** is a one-to-many, foreign key, attribute pointing to the simulation model running a current game.

## MarketAttributeType Model

This model stores a parameter associated with a team’s product.



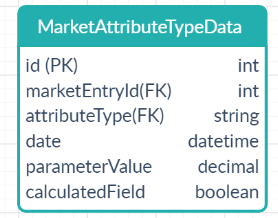
The **label** attribute stores the name of the parameter. The requirements specification has identified that these parameters will be:

* Price
* Market Share
* Revenue Share
* Sales Units
* Sales
* Cost of sales
* Overheads
* Net Profit
* Net Margin
* BE Price

Note: Some of these attributes (Price) can be set by the user, others can only be edited by the system.

## MarketAttributeTypeData Model

This model stores the datapoint for the MarketAttributeType.



The **marketEntryId** is a one-to-many (one MarketEntry, many MarketAttributeTypeData) foreign key. This maps the datapoint to the team that it belongs to via its market entry.

The **attributeType** is a one-to-many (one MarketAttributeType many MarketAttributeTypeData) which maps the datapoint to the market attribute.

The **date** attribute stores the date of entry for the datapoint. It is important to store the date for each datapoint so visual analytics such as graphs can show changes in the metric over time by plotting the collection of datapoints for the given market attribute for the given team. An example of this would be showing a graph of profits throughout the competition (Graph of profit against time). This is only possible if the attributes calculated/set for each day are stored.

The **parameterValue** attribute stores the value of the attribute. The data type of this field is decimal as it allows for real numbers to be stored: percentages for market share, currency for price and profit, and integers for unit sales.

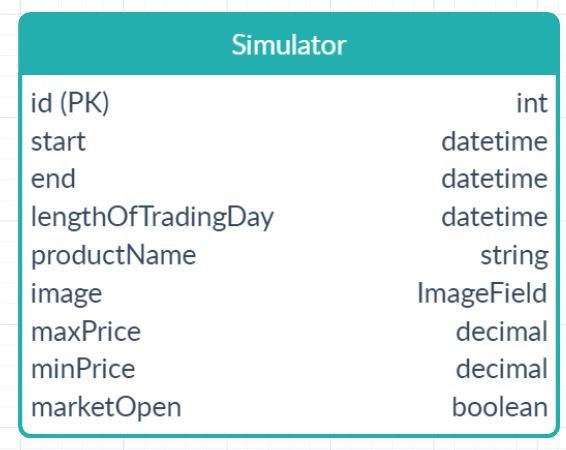
The **calculatedField** attribute is a logical flag indicating whether the field is calculated by the system or whether it is editable by the user. This will allow for the system to realise that the user can edit the Price field and that the user can only read the other attributes which get automatically calculated and updated.

***Note****:* The system calculating each of the parameters will need to be aware of what MarketAttributeType object they are to output their data to. For example, the method calculating the market share or a given teams product will:

* *need to perform a lookup by label and find the MarketAttributeType object who’s label equals ‘Market Share’.*
* *create a new MarketAttributeTypeData object for this MarketAttributeType for the team it has performed the calculation for*
  + *set marketEntryId to the teams MarketEntry object*
  + *set the attributeType to be the MarketAttributeType corresponding to ‘Market Share’ object*
* *store this result of the ‘Market Share’ calculation as the calculatedField*

# Simulator Model

This model is the driver containing key game parameters.



The **start** attribute stores the date time that the game starts.

The **end** attribute stores the date and time that the game ends.

The **lengthOfTradingDay** attribute determines how long a trading day lasts. It must be less the time than the length of the start date to the end date.

The **productName** attribute determines the name of the product that will be sold during the game. This must be changeable by YES staff between games.

The **image** attribute is used to show an image of the product. This is optional as It will be the same image displayed to all users since all users have the same product. If implemented, this will be set by YES Staff.

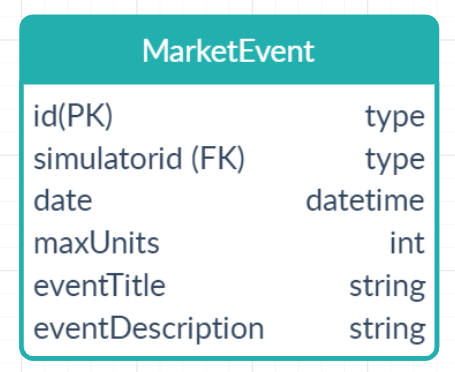
The **maxPrice** attribute specifies the maximum price that can be set for the product during the game.

The **minPrice** attribute specifies the minimum price that can be set for the product during the game.

The **marketOpen** attribute is a logical flag which signals whether the trading day is open or closed. Team actions such as editing strategy and price can only take place when the market is open (marketOpen = true). This means that certain actions should check whether the market is open before carrying out the action. If the market is closed i.e, the system is calculating sales for teams and updating their MarketEntry attributes, then the teams should be notified that their request cannot be processed at that time.

# MarketEvent Model

This model specifies special events which change market behaves during the game.



The **simulatorid** attribute is a one-to-many (one Simulator many MarketEvents), foreign key, which states what simulator the MarketEvent will affect.

The **date** attribute is the date that this event will take place. The form should allow the YES staff member to select an event that spans multiple days. The backend should then calculate how many days the event spans and add an identical event object for each day. When the market closes at the end of the trading day. The method allocating sales will check for any MarketEvents taking place that day and factor the first one it finds into the calculation.

The **maxUnits** attribute determines allows for a cap to be sent on the number of sales given that day. This could allow for a pandemic or holiday to be simulated where there is a sudden drop or spike in sales.

The **eventTitle** attribute is used to communicate to the teams the event that is taking place. This allows for them to adjust their approach if they choose to do so. The title may be ‘National Lockdown Declared’. This would then display on the team’s dashboard.

The **eventDescription** attribute accompanies the eventTitle attribute. It provides additional details about the event. For the ‘National Lockdown Declared’ example, the additional information may be ‘Shops are required to shut from 8pm-7am and members of the public cannot stay in cafes and restaurants for more than 1 hour’. This would allow teams to consider the situation and discuss a valid adjustment in product strategy which will minimise damage to their ranking.

# PolicyEvent Model



The **policyid** attribute is a one-to-one (one PolicyEvent to one Policy) foreign key which points to a Policy object.

The **marketeventid** attribute is a one-to-many (one MarketEvent to many PolicyEvents) foreign key which points to the market event that the PolicyEvent belongs to.

The weightingFactor attribute determines the new value of the Policies weightingFactor for the given MarketEvent.