CS261 Planning & Design Report
Group 24

Stylianos Maimaris Tim Marshall-Cox Maciej Lulka Folabi Ayonrinde Josh Adams

Lewis Buttle

1 Preamble

This document decomposes the system to be designed, detailing and justifying every aspect and sub system, provides descriptions of the database and classes to be utilised, and outlines the testing procedures to be implemented, to ensure the system meets or exceeds the requirements stated in the previous document. The diagrams used are as presented in the CS261 System Design slides [1]. The Planning and Design report is targeted towards the client (Deutsche Bank) and the module organiser (Dr. James Archbold).

2 Group Plan & Scheduling

2.1 Utilities

The team chose to use a private GitHub repository for sharing and working on the project implementation. For documentation, the reports are to be written in LaTeX using Overleaf and additional documents/drafts and ideas will be shared using Google Drive. Further communication (voice chat) will take place using a private Discord server.

2.2 Scheduling

Because of the nature of the project (having two strict deadlines for each deliverable), the team has chosen to organise its deadlines as shown in Figure 1. Each deadline is expected to take less time than what is allocated (an extra day is allocated for each task that is part of the requirements and design reports and 2 days for implementation and final report). This is done on purpose to ensure all aspects of the project run on time.

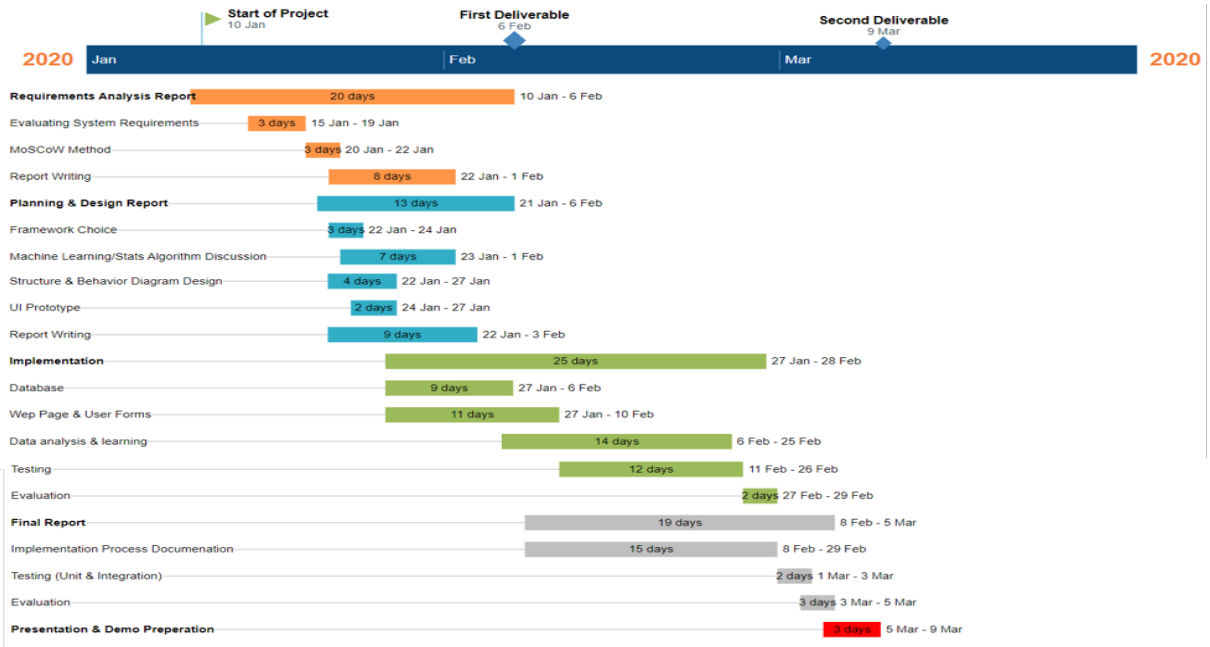

Figure 1: Gantt Chart: Project schedule and main tasks

3 Software Development Methodology

The team has chosen to adapt a mixture of plan-driven and agile approaches for the development of the system. This was done, in order to accommodate for the strict deadlines associated with the project for the requirements and design documents (plan driven) but to also allow for changes to the set of requirements based on the implementation. After the delivery of the reports mentioned previously, the team will use an agile, scrum based approach. Sprint cycles will have a one week duration and start at the beginning of each week. Besides weekly meetings, the team will utilise a Discord server for daily discussions of current work. Finally, customer feedback will be provided by the customer through the CS261 Forum.

4 Technical Description

On a technical level, the purpose of this report is to detail the design and planned implementation of a solution: to validate, identify, and subsequently correct mistakes within the dataset of a derivative (a contract that derives its value from an underlying entity, a speculative purchase) given by a user. Validated data will then be stored and used by the system to inform later validations - as though the system will 'learn' from the data. The system will exist to minimize accidental errors; of any kind, when documenting a derivative. Moreover, this document will aim to decompose the problem into a planned approach using clear, detailed plans and designs, and also breaking down the main problem into atomic solvable functions.

5 Framework Choice

As stated in the requirements analysis document, the system will be implemented through a web application. The team has decided to develop the system using the Python-based free and open-source web framework Django [2]. The choice was made as Django allows for rapid development, which is essential for this project. Furthermore, it integrates a database system using an object-relational mapper which allows the schema/layout to be written in the form of Python classes. Lastly, since the team is expected to utilise a machine learning and statistical analysis approach for detecting and correcting errors, utilising a Python-based framework was very important. A general overview of the Django web app is shown in Figure 2.

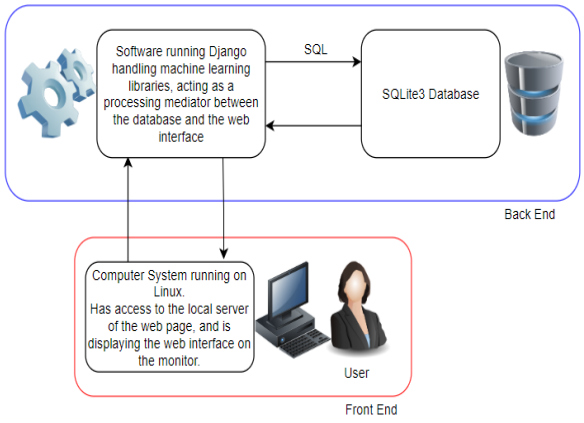

Figure 2: General System Architecture

5.1 Back-end

5.1.1 Programming Language Choice

As with any Django based web application, back-end processing will be implemented using Python3. Specifically, statistical analysis of data will be used to implement the machine learning model, by using Django's class/function based "view" system. This will allow for real time processing of user input. In addition, Django integrates a database as part of the web application. By default, Django uses SQLite3, which meets the system's storage and querying needs.

5.1.2 Database

Figure 3 highlights the breakdown of the database into its constituent tables. Every trade will be represented by a record in the derivativeTrades table, with foreign keys to records in subsequent tables. The most up to date currency values and product/stock prices will be read into the corresponding tables on a daily basis, and so all derivative trades from a single day will all use the most up to date values that exist for these attributes (no foreign key relations are used for these attributes as the data will be read from these tables and added to each trade separately). The archived field in the derivativeTrades table is a Boolean flag that determines whether the trade has expired (twenty four hours have passed since its creation) and so its ability to be edited upon and/or deleted has likewise terminated. Company codes store all companies listed that can trade, and this will be editable by the user, who will be able to add new companies in as the system exists. Similarly, they will be able to add new products and their prices into the ProductPrices table.

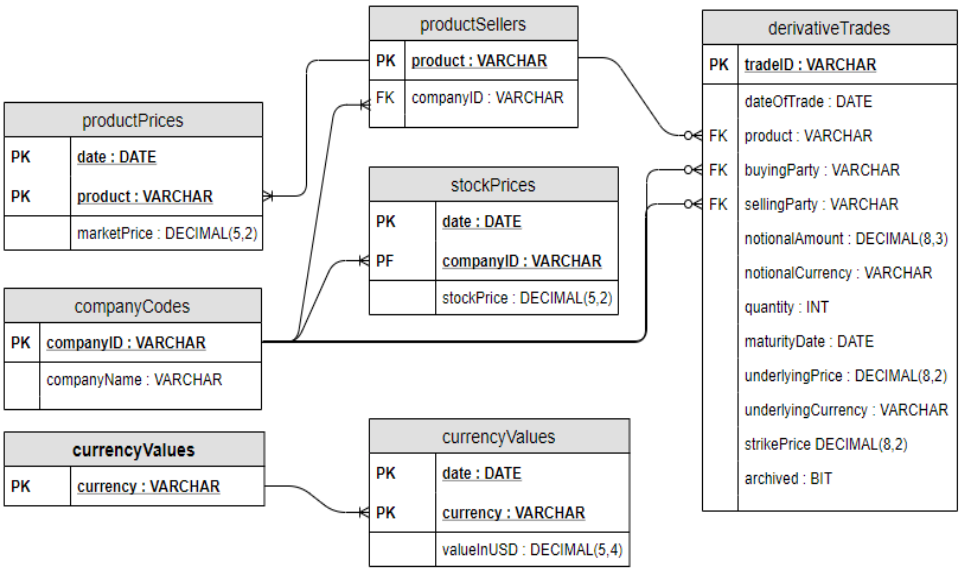

Figure 3: Database Entity Relationship Diagram

5.1.3 Machine Learning & Data Analysis Model

A statistical analysis approach has been chosen for our error detection subsystem to be implemented with the help of the scikit Python library [3]. Such a choice will ensure an efficient way of detecting erroneous data without the need to resort to creating artificial invalid data. Were such an approach to be taken, this could lead to biases meaning the deep-learning subsystem could be really effective when dealing with certain errors is has been trained on. However, as soon as it encountered an unknown error, it would be impossible to be certain of its behaviour, as is the nature of such methods.

The linear regression approach enables flagging of any error that does not match previous data, regardless of the type of error, without requiring specific training data to portray that particular kind of error. For each new trade two lines of best fit for the current data shall be calculated - one of unit-price against time, and the other of quantity purchased against time. These will be used to calculate the residual standard deviation of the dataset for each of price and quantity (where the dataset constitutes of all previous trades of this particular derivative in the past 10 years). To calculate the line of best fit, the least squares method will be used [4, 5].

$$m = \frac{\sum_{i=1}^{n}(x_i - \overline{X})(y_i - \overline{Y})}{\sum_{i=1}^{n}(x_i - \overline{X})^2} \qquad b = \overline{Y} - m\overline{X}$$

This results in the equation of a line: $y = mx + b$ where x is the time of the trade, and y is the cost per unit. With this, given a time x, the cost y is estimated. Using the given dataset, a residual for each y value can be calculated. This is defined as the difference between the actual y, and the estimated y. With these residuals, we can calculate the residual standard deviation with the formula:

$$S_{res} = \sqrt{\frac{\sum(Y - Y_{est})^2}{n-2}}$$

This residual standard deviation will be used to define two thresholds (each as a multiple of the standard deviation). Any trade within the first threshold ($Y_{est} \pm$ threshold_one) will be deemed valid by the system. Any trade not within this interval, but within the second threshold ($Y_{est} \pm$ threshold_two) will ask the user for input correctness confirmation. Any data outside this second interval shall be automatically deemed incorrect and hence passed to our error correction algorithm. In the event that this algorithm cannot come up with a satisfactory solution, the user be informed of this and again asked for input correction or confirmation.

5.1.4 Automation

To automate corrections an algorithm will the following output shall be produced: It will manipulate the given data in a systematical manner and test each such attempt against the statistical model to establish if any of them deem likely solutions. The algorithm will test for, but is not limited to the following:

- Trailing zeros - If the data entered is significantly above or below the expected value, there may be an additional (or absent) zero at the end of the number.

- Fat-finger - If there are consecutive numbers within the number (i.e. 1459), the user may have accidentally pressed both keys at once. In this case, the algorithm should test 149, and 159.

- Double-presses - If the same digit is repeated within the number (i.e. 778), the user may have accidentally double pressed a key. The algorithm should test 78 to see if it is a better fit with previous data.

If these and other tests are not able to produce an entry that fits the data within acceptable thresholds, the user will be prompted to check the data before confirming that it is correct. Note: These tests will only occur when the entered data is sufficiently different from past data, as defined by the second threshold.

5.2 Front-end

5.2.1 Programming Language Choice

As the project will be developed using the Django framework, the team has decided to utilise the standards suggested by the Django documentation. Web pages will be created using the Django Template Language in conjunction with HTML. Doing so, will allow the team to create a single HTML file from which all other web pages can inherit from, ensuring consistency throughout the system. Furthermore, Django allows for the use of static files (of which CSS and JavaScript will be used) which will ensure the web page presented is easy to use and provides maximum functionality.

5.3 Relationship between systems

The main system is broken into a number of sub systems as shown in Figure 4. The system must allow new trades to be created, and derivative information entered by the user. This data is passed to the back end of the system where it is analysed against the machine learning components to determine whether any errors exist within the inputted trade. Appropriate functionality will occur if any errors are determined, and user input will be needed to verify the correctness of the decision. Over time this process will be automated and so abstracted away. All data is then saved to the database for further processing. Further functionality includes allowing the user to search through past trades, editing and deleting those within a recent twenty four hour window, and compiling a daily report of all trades occurred within a single day. These reports will also be searchable by date.

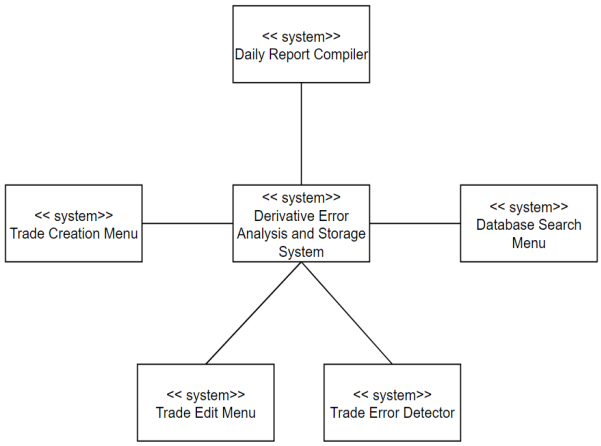

Figure 4: Context Model: Operational context of the main system and other systems

5.4 Definitions of entities

Django uses "apps" which act as separate components for the web application with each app having its own models and views. There will be a number of different Django models and views for this project, each handling a select functionality as is illustrated by Figure 5. The derivative model will be used to handle functionality based on a single derivative trade. Validation on the inputted attributes will be handled by the web implementation rather than being inserted within the derivatives model. Three further views will be created to handle each one of the following; creating a new trade and editing or deleting one. There will also be a view to handle creating daily reports which will be returned to the user as a PDF using Django's PDF library [6]. The back end will also be modularised into separate Django apps. The machine learning component will be included within its own app.

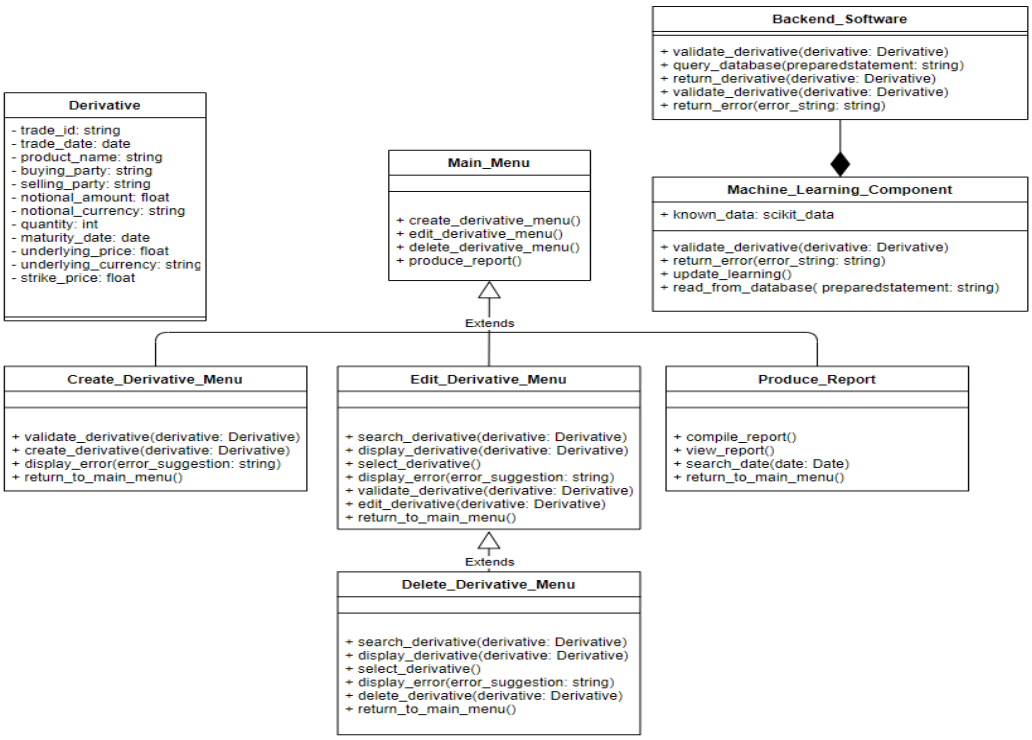

Figure 5: Class Diagram

6 Description of processes & activities

6.1 User Interaction

The main page of the web form will present the user with a number of different facilities(shown in Figure 6); creating a new trade and adding it to the database, searching for a trade and editing or deleting it and producing a daily report of all trades occurred within that day.

1. New Trade:
   - When a user clicks to create a new trade, they will be presented with a web form, allowing them to enter data for the following fields; trade ID (String), Product (String), Buying Party (String), Selling Party (String), Notional Amount (Double), Notional Currency (String), Quantity (Integer), Maturity Date (Date), Underlying Price (Double), Underlying Currency (String) and Strike Price (Double). Each of the above fields will have validation present to determine whether the input matches that of the correct data type. Any incorrect inputs will be flagged to the user and they will be unable to submit the form.
   - The underlying price entered by the user has to match the price stored in the database for that object on that day. When the form is submitted, a back end function is triggered which will run a query extracting the underlying price of the product from that day. If the record is not found, then the product string was entered incorrectly, and so the system will notify the user. If a value is returned, it is checked against the input, and if the user was incorrect then a notification will be presented to them. If the field is left blank, which is a valid operation, then the system will automatically fill in the price and submit.
   - The system will also check whether the entered buying and selling party codes are correct., by running a back end database query to see if they exist in the database. If not, then a message will be displayed to the user, telling them there was a mistake. They will be able to correct their mistake.
   - Similarly, values that are changing over time, such as currency, will also be checked if an input is given to ensure it runs correctly, and if left blank, them will be auto filled with the most currently up to date value retrieved from the table currencyValues.
2. Edit Trade:
   - The user will also be able to search for a trade within a twenty four hour window and make amendments to them. Any trades after this window has expired will be archived and non-mutable.

- The user will have the ability to search for a trade, off of a number of attribute fields. The user will select the field to search for from a drop down menu and then type the attribute into the search bar. Basic validation will be in place to ensure that if an integer/string is required then an integer/string has been entered. This will trigger a query that runs in the background, searching the database table derivativeTrades for the trade and then outputting the information onto the page. The user will be able to search via; trade ID, date of the trade, the product and the buying and selling parties.
- Once a user has found a trade, then they will be presented with the option to edit any of the fields. Validation, the same as creating a new trade, will be in place, to ensure that the correct input is entered into each individual field. The user will then be able to submit the new, updated form into the database. This will not reset any timers, and so the user won't have another twenty four hours to make amends. Once resubmitted, a database query will be run, accessing the derivativeTrades table and updating the values.
- The user will also have the ability to delete a trade, and if clicked, will delete the trade and it's record from derivativeTrades. This means that if any current daily reports exist, then it must be recompiled before being shown to the user in order to remove the erroneous trade.

3. Producing Daily Report:

   - When this option is clicked, a database query will be run and extract all trades from the derivateTrade table from within the last twenty four hours, converting them into a PDF format. This report will document a table displaying all attributes from the database, including trade ID, buying and selling parties, quantity and notional value. This report will be archived so that it can be searched from in future.

4. Searching Archived Reports:

   - The user will be able to search through the archive of generated reports based on date. The date will be validated to ensure it conforms to the date standard and is within the last ten years. A database query will then be run, searching the database table for the report, which will then be displayed to the user.

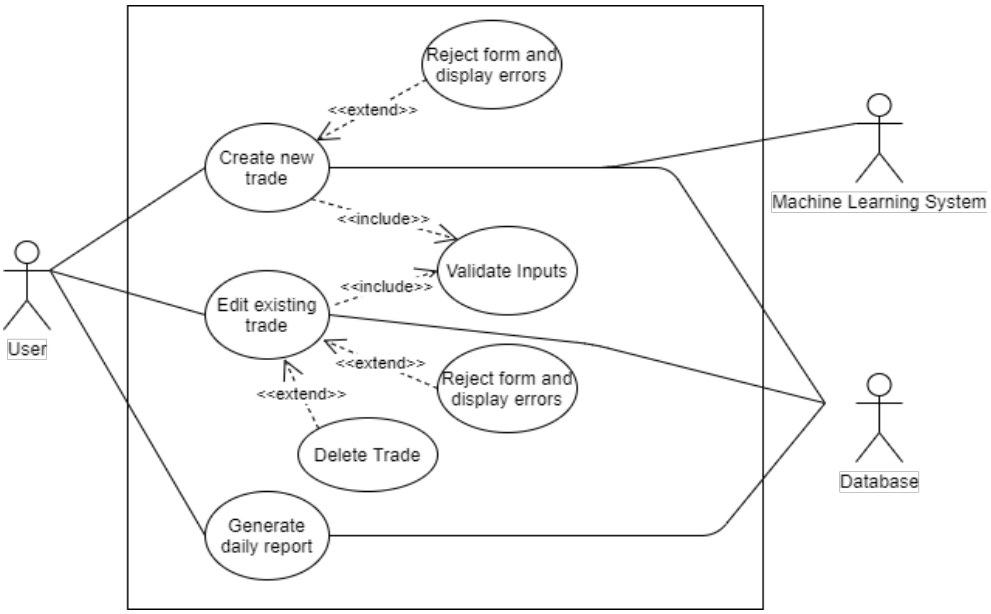

Figure 6: Use Case Diagram

6.2 Interaction between components (sequence diagrams)

Figure 7 shows the sequence diagram of the entire system. The user enters a new trade, and this is validated on the front end. The data is then sent to the machine learning component which checks for semantic errors, such as fat finger, which the front end validation is unable to pick up, using the learning capabilities of the system to use past experiences to educate future choices. If there are no detectable errors, the trade is saved, else the errors are flagged and passed to the automation system. This system is responsible for taking the errors and making judgements dependant on them and implementing autonomous corrections. If the user agrees with the changes then the trade is saved, and the automation system makes a note that it got the correct choice. If not, then it saves the user's overriding correction, and notes that it made the wrong decision.

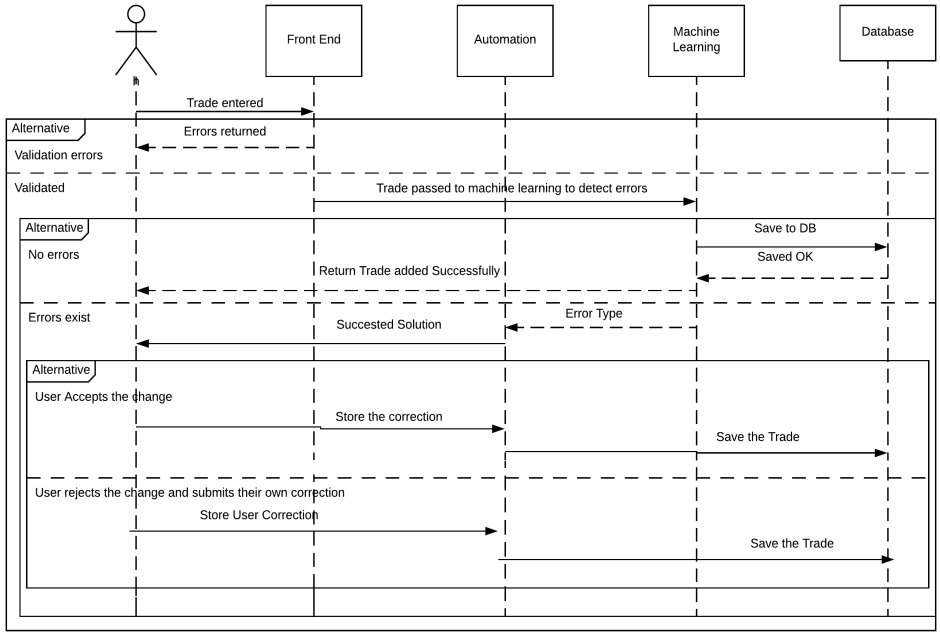

Figure 7: Sequence Diagram: from user input to input being stored in the database

7 User interface design

This section shows a set of prototype pages for the main menu (Figure 8), creating new trades (Figure 9), generating a daily report (Figure 10), editing and deleting trades (Figures 11 and 12 respectively) with which user will be expected to interact with. They were created in order to give a clear guide to the front-end team when designing the user interface.

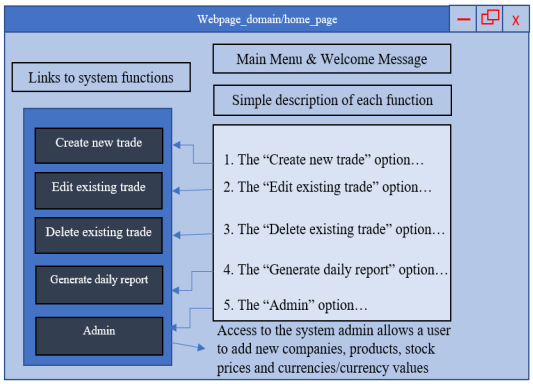

Figure 8: Main Menu Prototype

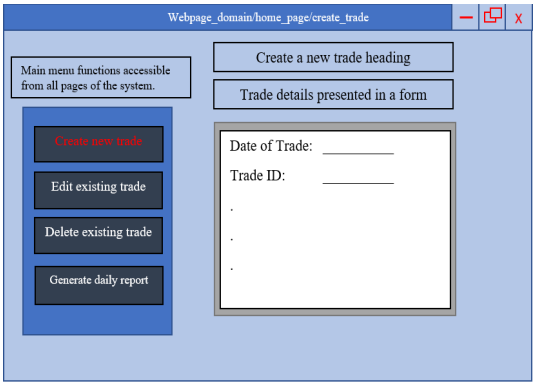

Figure 9: Create new trade Prototype

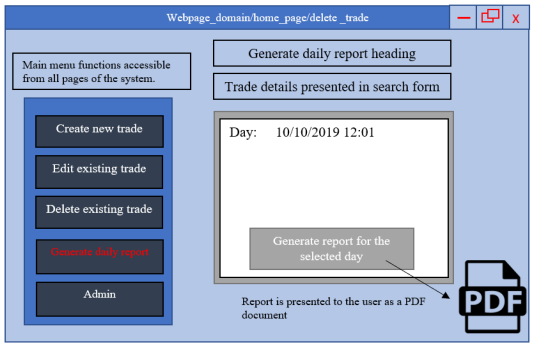

Figure 10: Generate daily report Prototype

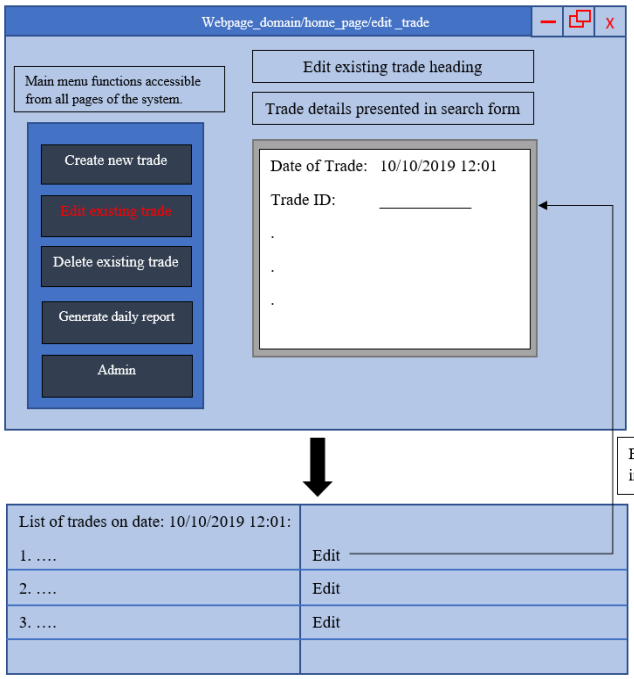

Figure 11: Edit existing trade Prototype

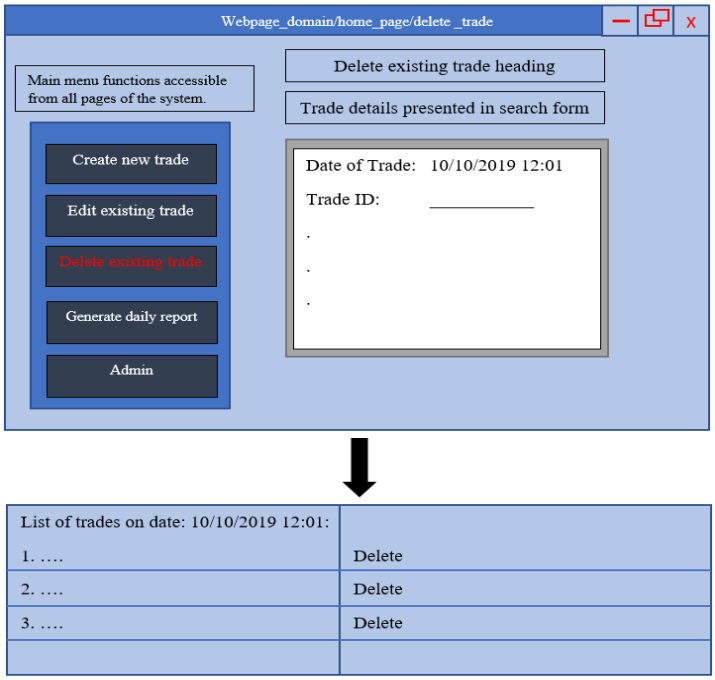

Figure 12: Delete existing trade Prototype

8 Test Plans

1. Web Form:
   - **[Requirement 1]** There must be an entire web form application created, which would allow the user to select between creating a new trade, editing and deleting existing trades, and generating a daily report.
2. New Trade:
   - **[Requirement 2]** The user must be able to create a new trade.
   - **[Requirement 6]** Validation on each input; string only fields will be tested to determine whether they allow numeric characters, alphanumeric fields will be tested to determine if they allow both all string, all numbers and mixes of both, and finally numeric fields will be tested to ensure they allow numbers entered of the correct form (i.e. integer of float only)
   - The incorrect underlying price will be entered, to ensure the system is capable of detecting this, and a message should be returned to the user with the correct price.
   - Product ID's should be checked to determine whether the product being traded exists, and if incorrect, then appropriate messages should be sent to the user.
   - Buying and Selling ID's should be checked to determine if the companies exist, and if they don't the system should flag this as an error and note the user.
3. Edit/Delete Trade:
   - **[Requirement 3/4D]** The trade system will be checked to ensure it allows all trades from within a 24 hour window (exclusive) to be edited. Tests will be performed on trades entered 23/24 hours ago, and the system should handle the correct dates appropriately.
   - **[Requirement 3/2D, 4/2D]** The system should allow a number of attributes to be entered to search by, including date, trade ID, product ID and buying and selling parties.
   - There should be appropriate validation when entering the attributes to search by, and if incorrect the search should not occur.
   - **[Requirement 3/3D]** Each field within the trade should be editable for the user, and the appropriate validation should still exist per attribute. The same validation tests will be performed as the was on the new report.
   - **[Requirement 3/4D]** The time limit for a trade should not change after submitting changes, and so a trade with a limit of an hour remaining should not be reset to a fresh twenty four limit after editing.

- All changes made to a trade should persist and be entered into the database, and then returned the next time the form is selected, even if a form is edited and then researched for.
- If a trade is deleted, then it should not be possible to search for it again. It should also be removed from any stored, compiled, daily reports.
- [**Requirement 4/3D**] Trades will not be able to be deleted after twenty four hours from their creation.

4. Produce Daily Report:

- [**Requirement 5/1D, 5/2D**] The user should be able to compile the daily report, and it should return a generated PDF with all the trades occurring from that day.
- There will be tests in place to ensure the report contains only trades occurred on that day, from 12 AM onwards. Any deleted trades should not be shown. All edits must be up to date.
- [**Requirement 4D**] The user should be able to search through the archive of generated reports.

9 Risk register

Potential risks have been identified and documented as shown in Figure X. The risk register[7] [8] is used to list the most important risks that have been identified and analyse their potential impact given the probability of occurring and the loss of size (in days) if they occur. In addition, each task has a mitigation in place which is then used to estimate the new loss of size. Finally, the total risk exposure(sum of exposure per risk) is calculated before and after mitigations.

Risk	Probability of occurance	Loss size (days)	Risk exposure (days) (probability x loss size)	Mitigation	Loss size after mitigation	Exposure after mitigation
Inability to verify error detecting model's effectiveness before full implementation	5%	15	0.75	Extensive unit and integration testing for error detecting model	4	0.2
Erroneous dataset created by the team for testing is insufficient	10%	2	0.2	Multiple datasets per test will be created and tested	1	0.1
Uneven distrubution of workload (not enough man power in front-end/back-end)	15%	2	0.3	All team members will be familiar with the Django framework and associated libraries	1	0.15
Team member illness	20%	2	0.4	Bus factor of at least two	1	0.2
Poor time management/not meeting deadlines	25%	2	0.5	Gantt chart ensures there is a buffer for all deadlines	0	0
Short falls of machine learning library "scikit"	30%	3	0.9	Use backup library "numpy"	2	0.6
Inadequate user feedback (from forum)	40%	2	0.8	Feedback through module organiser and tutor	1	0.4
Total Risk Exposure			3.85			1.65

Figure 13: Risk Register of top 7 most risks and their mitigations

10 Extensibility

As the system will be developed using Django the system will be very easily extensible due to the framework's scalability. This is the case as Django uses a "shared-nothing" architecture which enables the addition of hardware at any level – database servers, caching servers or Web/application servers. Furthermore, the framework cleanly separates components such as its database layer and application layer. In addition, the system is expected to be very robust as unpredictable/invalid input will be checked both in the front-end and back-end.

11 Reliability

The front end of the system deals mainly with validation, and so this is exhaustive, and every possible input will be tested to ensure the code deals with it appropriately. This means that it won't be possible for a user input to break the code, and the system will always take the correct response to deal with any input. Database and back end functionality will be unit tested, and only ever called with correct inputs, as validated by the front end, and so there will never be any instances for the system to break.

12 Correctness

The design of the system proposed in this document if implemented fully, should meet or exceed the customer's requirements as each can be traced throughout the report. The requirements were used to design the document, and create the test cases, and so it follows from this that the system should perform the correct functionality, and after extensive testing, execute without fault.

13 Compatibility and Portability

The system is expected to be easy to install and maintain as the only requirements are Python3.6 or greater and Django 3.0.3 or greater. The installation process will be documented to ensure all steps are clear. Furthermore, the customer should only ever interact with the system through a web browser. Lastly, being a web application the system should be accessible from most devices which support some form of web browser.

14 Modularity and Reuse

Modularity will be ensured through Django's view/template system, which will act as a means of reusing modules of code. Module code will be used providently to better organize group work, and to make it easier to assign coding tasks to team members with minimal confusion (by atomically breaking down problems). Modules will handle derivatives around the system, specifically the module of querying for a derivative in the database will be reused many times during development, as it will be used for finding derivatives wanting to be edited or deleted, as well as by the machine learning code when it needs to update itself. Hence, this module will use prepared statements to be called repeatedly using the derivative data being passed around the system to access the database. Similarly, a module to insert and delete from the database using prepared statements will be used as well.

15 Security

Although security is not something the team will focus on as there are no such requirements for the system, Django's built in and on-by-default security features ensure that the current solution will be able to incorporate more powerful security measures in the future.

16 Fault-tolerance

Since the system will function through a web server, there is no client-side dependencies for storing data. However, the machine hosting the system is expected to be backed up daily to ensure that total hardware failure does not lead to loss of data.

References

[1] J. Archbold, "Topic 4: System design 2," Feb 2020. [cited 01 February 2020]. Available from: https://warwick.ac.uk/fac/sci/dcs/teaching/material/cs261/l7-1.pdf.

[2] "Django documentation," Feb 2020. [cited 02 February 2020]. Available from: https://docs.djangoproject.com/en/3.0/.

[3] "scikit-learn: Machine learning in python," Feb 2020. [cited 01 February 2020]. Available from: https://scikit-learn.org/stable/.

[4] "Line of best fit (least square method)," Feb 2020. [cited 01 February 2020]. Available from: https://www.varsitytutors.com/hotmath/hotmath_help/topics/line-of-best-fit.

[5] W. Kenton, "Residual standard deviation," Feb 2020. [cited 01 February 2020]. Available from: https://www.investopedia.com/terms/r/residual-standard-deviation.asp.

[6] "Outputting pdfs with django," Feb 2020. [cited 01 February 2020]. Available from: https://docs.djangoproject.com/en/3.0/howto/outputting-pdf/.

[7] T. Arnuphaptrairong, "Top ten lists of software project risks: Evidence from the literature survey," Feb 2020. [cited 02 February 2020]. Available from: http://www.iaeng.org/publication/IMECS2011/IMECS2011_pp732-737.pdf.

[8] "Software development risk management plan with examples," Feb 2020. [cited 02 February 2020]. Available from: https://www.castsoftware.com/research-labs/software-development-risk-management-plan-with-examples.