Informatics Institute of Technology

In Collaboration With

The University of Westminster, UK

The University of Westminster, Coat of Arms

A Novel Approach to Time Series Forecasting using Liquid Time-constant Networks

A Design by

Mr. Ammar Raneez

W1761196 | 2019163

Supervised by

Mr. Torin Wirasingha

November 2022

This Project Proposal is submitted in partial fulfilment of the requirements for the BSc (Hons) Computer Science degree at the University of Westminster.

Contents

List of Tables i
List of Figures ii
1. INTRODUCTION 1
2. DESIGN GOALS 1
3. HIGH LEVEL DESIGN 2
4. SYSTEM DESIGN 4
4.1. Choice of design paradigm 4
4.2. Data flow diagram 4
4.3. Algorithm design 6
4.3.1. Existing LTC architecture 6
4.3.2. Algorithm proposed by the author 6
4.4. UI design 10
4.5. System process flow chart 10
REFERENCES I
APPENDIX I – Algorithm Intuition II
APPENDIX II – UI Wireframes III

List of Tables

Table 1: Design goals of the proposed system 1

List of Figures

Figure 1: Three-tiered architecture (*Self-Composed*) 2
Figure 2: Data flow diagram - level 01 (*Self-Composed*) 5
Figure 3: Data flow diagram - level 02 (*Self-Composed*) 5
Figure 4: System process flow chart (*Self-Composed*) 10
Figure 5: Algorithm intuition (*Self-Composed*) II
Figure 6: UI wireframes – Home (*Self-Composed*) III
Figure 7: UI wireframes – News (*Self-Composed*) III
Figure 8: UI wireframes – Cryptocurrencies (*Self-Composed*) III
Figure 9: UI wireframes – Cryptocurrency (*Self-Composed*) III
Figure 10: UI wireframes - Admin login (*Self-Composed*) IV
Figure 11: UI wireframes – Admin model configuration (*Self-Composed*) IV
Figure 12: UI wireframes - Forecast (*Self-Composed*) IV

Acronyms

AI	Artificial Intelligence.
API	Application Programming Interface.
AD	Automatic Differentiation.
ARIMA	Autoregressive Integrated Moving Average.
BPTT	Back-Propagation Through Time.
BTC	Bitcoin.
CT-GRU	Continuous-time Gated Recurrent Unit.
CT-RNN	Continuous-time Recurrent Neural Network.
DL	Deep Learning.
GPU	Graphics Processing Unit.
LSTM	Long Short-Term Memory.
LTC	Liquid Time-constant.
ML	Machine Learning.
(s)MAPE	Symmetric Mean Absolute Product Error.
MASE	Mean Absolute Scaled Error.
MSE	Mean Squared Error.
N-BEATS	Neural Basis Expansion Analysis for interpretable Time Series
NLP	Natural Language Processing.
ODE	Ordinary Differential Equations.
POC	Proof-Of-Concept.
REST	Representational State Transfer.
RMSE	Root Mean Squared Error.
RNN	Recurrent Neural Network.
SDE	Stochastic Differential Equation.
SGD	Stochastic Gradient Descent.
TS	Time Series.
UI	User Interface.

1. INTRODUCTION

In this chapter, the author focuses on selecting suitable architectural structures for implementation, considering the gathered requirements. Specifically, high-level, low-level, and associated design diagrams are presented alongside necessary UI wireframes and the reasoning behind each choice.

2. DESIGN GOALS

Table 1: Design goals of the proposed system

Goal	Justification
Performance	A typical flow in TS forecasting requires retraining the model whenever a prediction is made, as the data the model had been trained on could be outdated. However, as multiple features are being used in the proposed system, this can severely hinder performance. The author can avoid this by storing past data and only fetching needed data when necessary; as a further step, the data can be fetched periodically. The model can automatically be retrained beginning each day (which would deem the retraining step on each inference unnecessary) as the solution proposed.
Usability	Based on the analysis obtained during the requirement-gathering phase, there were mixed thoughts on whether the application would benefit people who are not experts in cryptocurrencies. Therefore, this requirement is mandatory as it is crucial to create a system that is as user-friendly as possible to be used by users across all levels of expertise.
Quality	The output must be of the highest possible quality. Also, as identified in the gathered requirements, the system must display a range of prices to provide more conviction. Additionally, providing insights into how the model made the inference is an added benefit if time permits.
Maintainability	As implied by the author, the research must yield two products for the project to be successful. The goal of maintainability is solely for the research product proposed. The architecture of the algorithm must be optimal and independent to be able to be used as a reference for future research.

3. HIGH LEVEL DESIGN

The system's high-level architecture design is depicted below. The author chose a three-tiered architecture because of the distinct separation of concerns of the presentation, logic and data layers.

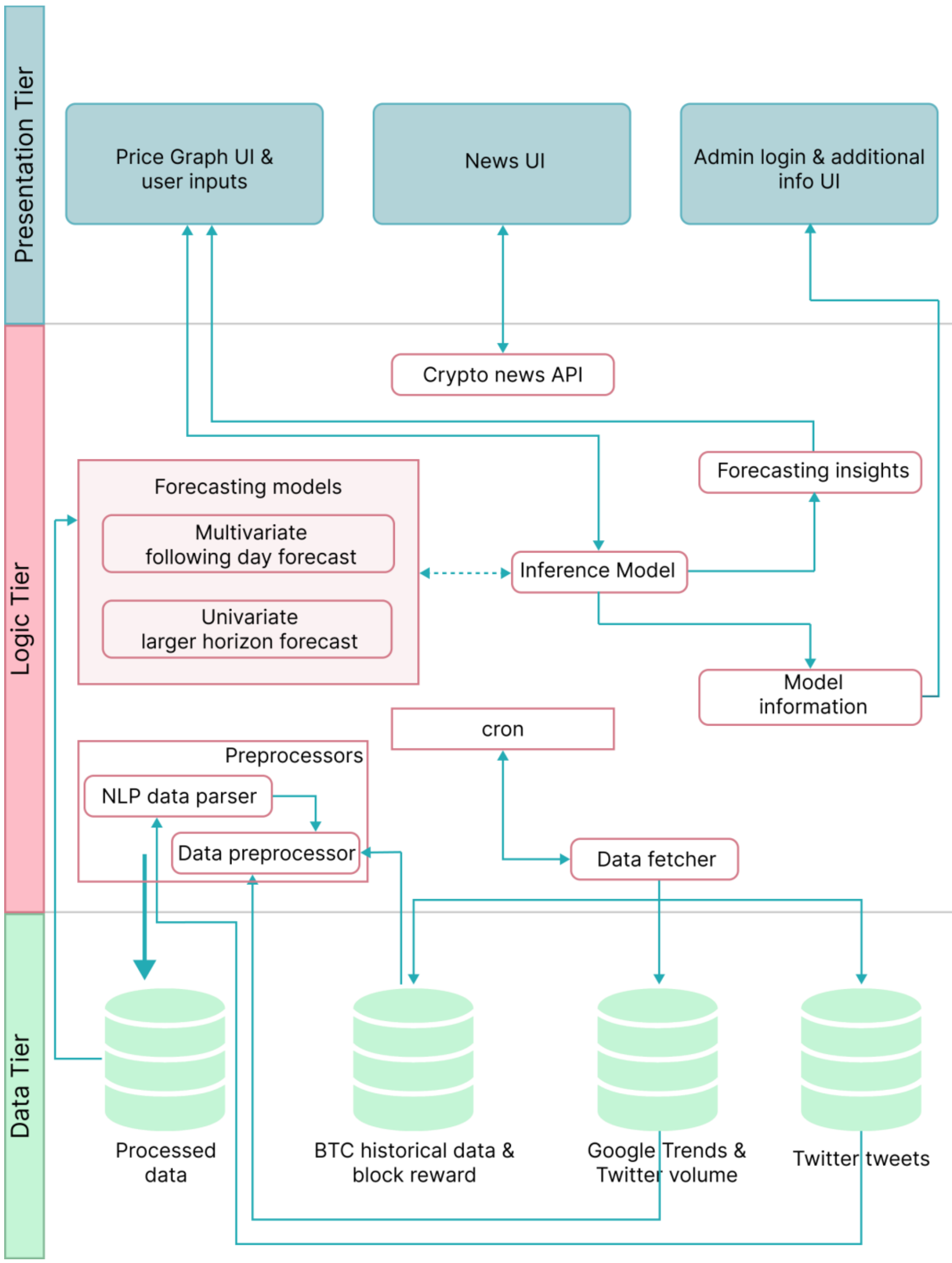

Figure 1: Three-tiered architecture (*Self-Composed*)

Data Tier

All data in this layer are fetched from an API and stored in individual documents to ensure updated data is available whenever necessary.

- BTC historical data & block reward – historical data of BTC closing prices of the past several years and the associated block reward obtained for mining BTC.
- Google Trends - historical data of the number of searches made each day that are BTC related.
- Twitter volume – historical data of the number of tweets posted each day that are BTC related.
- Twitter tweets - historical data of the tweets posted that are BTC related.

Logic Tier

The logic tier consists of the base logic performed on the data in the data tier to provide an output in the presentation tier.

- Preprocessors – consist of code required to process the raw data fetched from the API's so that the forecasting model can use it.
    - Data preprocessor – required for general preprocessing steps such as normalization and cleaning of data.
    - NLP data parser – required to perform sentiment analysis on the tweet data and named entity recognition to give more weightage to specific tweeter's sentiment.
- Data fetcher & cron – the automated scheduler that the script will run periodically to ensure that the data and model are up-to-date.
- Forecasting models – models that will be used to provide forecasts.
    - Multivariate following-day forecast – utilized for the following day forecasts.
    - Univariate greater horizon forecast – utilized for forecasts requested for days ahead of the following day.
- Model information – extra information of the model that the admin could view (ex: accuracy, no. of epochs).
- Forecasting insights – additional information presented to the user to demonstrate forecasting-related Explainability.

- Crypto news API – an additional third-party API to provide users with daily news about cryptocurrency.

Presentation Tier

The point of interaction where the user interacts with the system.

- Price graph UI & user inputs – main UI of the MVP that is presented to the user. It would display the current pricing graph, provide the user options to choose a future date, and generate a new chart with the inference.
- News UI – a minor sub-feature that will display news about the cryptocurrency world.
- Admin login & additional info UI – a 'could have' feature that will provide an authorized user to obtain information about the current model in use and, further, provide the ability to retrain the model by adjusting hyperparameters in use.

4. SYSTEM DESIGN

4.1. Choice of design paradigm

As identified in previous chapters, the choice of design paradigm is SSADM. To re-elaborate, as this research is primarily focused on developing a novel architecture with a novel algorithm, extensive experimentation is paramount. Furthermore, the selected programming languages do not promote OOP; instead, they encourage using function-based modules and components.

4.2. Data flow diagram

The data flow diagrams are depicted using level 0, level 1, and level 2, where level 0 is the context diagram presented in the SRS chapter.

Level 01

The level 01 diagram is an extensive breakdown of the core components proposed in the context diagram.

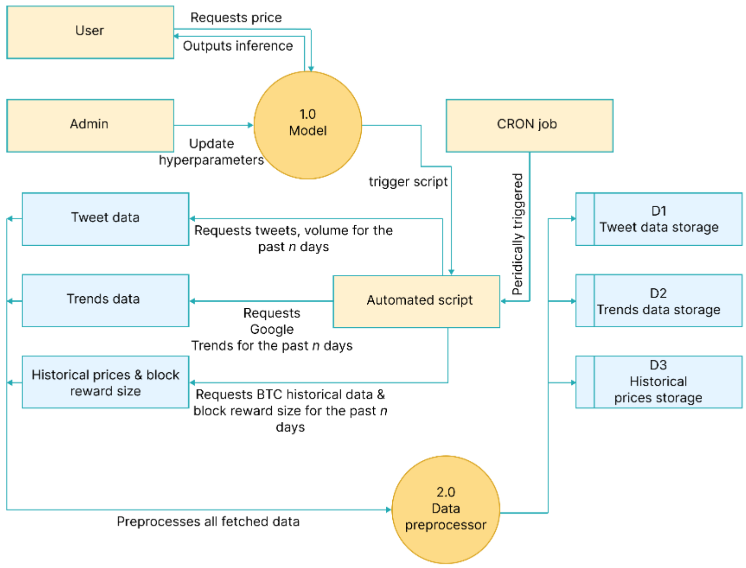

Figure 2: Data flow diagram - level 01 (*Self-Composed*)

Level 02

The level 02 diagram is a more extensive breakdown of the core data preprocessor component proposed in level 01.

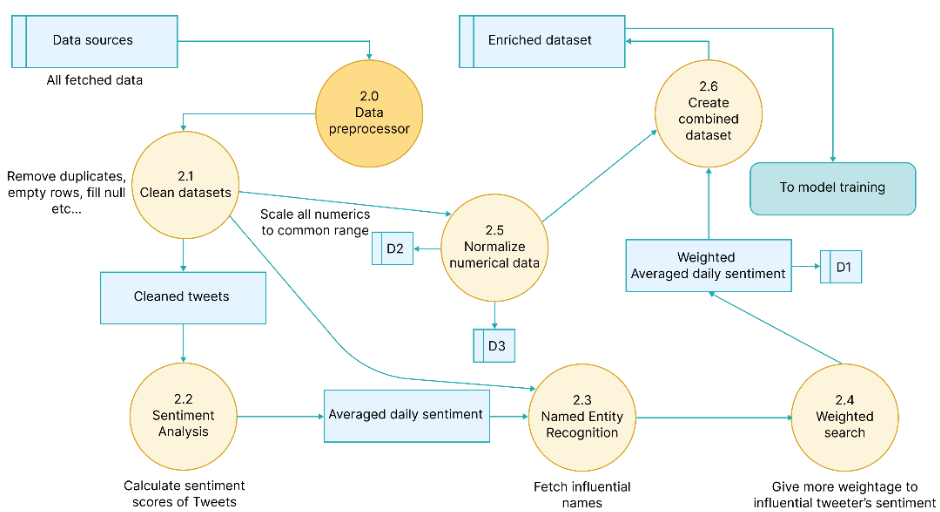

Figure 3: Data flow diagram - level 02 (*Self-Composed*)

4.3. Algorithm design

Upon gathering requirements to implement the research component, the author realized they could further enhance the existing LTC architecture by integrating flexible latent SDEs instead of the current ODEs. The author will therefore attempt to design and evaluate a novel algorithmic implementation inspired by the original LTC proposed by Hasani et al. (2020), which can be considered as their primary contribution to the body of knowledge. A simple illustration is available at **APPENDIX I** to gather intuition.

4.3.1. Existing LTC architecture

$$\frac{dx(t)}{dt} = -\left[\frac{1}{\tau} + f(x(t), I(t), t, \theta)\right] x(t) + f(x(t), I(t), t, \theta)A$$

τ	*Time-constant*
$x(t)$	*Hidden state*
$I(t)$	*Input*
t	*Time*
f	*Neural network*
θ, A	*Parameters*

The above formulation was proposed by Hasani et al. (2020), where a system of linear ODEs is used to declare the flow of the hidden state; the ODEs are of the following form.

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t)$$

Where S(t) represents the following nonlinearity

$$S(t) = f(x(t), I(t), t, \theta)(A - x(t))$$

The equation manifests by plugging the above equation into the system of linear ODEs.

4.3.2. Algorithm proposed by the author

Upon studying the abovementioned architecture, the author could utilize a linear system of SDEs to declare the flow to manifest a potentially novel algorithm with more flexibility for instantaneous adaptation of tiny changes. Moreover, this is an excellent enhancement as the additional component being developed belongs to the open market, which can have small instant price changes.

Formulation

Step 01 – transitioning from an ODE to an SDE

In simple terms, an SDE is an ODE with additional noise added at each step, which the model can use to model uncertainty.

$$Assume\ an\ ODE\ is\text{:}\ \frac{dx}{dt} = f(x); which\ obtains\ the\ expected\ slope\ of\ x(t)$$

The above ODE can be used to calculate the 'expected' slope, whereas the 'realized' slope differs from the 'expected' due to random noise, also called random Gaussian perturbations or Gaussian white noise. With that in consideration, the following can be derived:

$$An\ SDE\ is\text{:}\ \frac{dx}{dt} = f(x) + \mathcal{E}_{t+dt}; where\ \mathcal{E}_{t+dt}\ is \sim N(0,1)$$

$$Where\ N(0,1)\ is\ a\ Gaussian\ 0{,}1\ random\ variable$$

However, noise can be of varying intensities (some could be high, some could be low). Considering this varying intensity, the SDE can be further expressed as follows:

$$\frac{dx}{dt} = f(x) + g(x) * \mathcal{E}_{t+dt}; where\ g(x)\ is\ the\ intensity$$

As implied, the missing factor in the existing architecture that consists of ODEs is the absent stochastic transition dynamics (i.e., a noise for each timestep – which is vital to model the tiny unobserved interactions). The above equation considers the small unobserved interactions and uncertainties that could occur; this is further important in the context of TS data, as the initial state of data is unlikely to be certain.

Step 02 – adding neural networks into SDE dynamics

Based on the findings of Duvenaud (2021), the noise mentioned in the previous step can be considered as Brownian motion, a generalized form of the Gaussian noise. Researchers can produce the following by plugging Brownian motion into the equation determined in the previous step.

$$dx = f\big(x(t)\big)dt + \sigma\big(x(t)\big)dB(t)$$

A neural network can be integrated into the above equation to solve the system, resulting in the following equation:

$$dx = f_\theta(x(t))dt + \sigma_\theta\big(x(t)\big)dB(t)$$

$$where\ f\ is\ usually\ a\ tiny\ neural\ network\ and\ \theta\ are\ its\ parameters$$

Step 03 – Integrating the above equation into the LTC architecture

Moving back to the main problem at hand, the author can now construct a new formula by using the equation determined in the previous step.

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t)$$

As the above equation is a linear system of ODEs initially proposed by Lapicque (1907), the author could add the uncertainty noise to the equation to produce the following:

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) + \sigma\big(x(t)\big)B$$

The above equation now defines a stochastic process instead of deterministic evolution. Therefore, researchers can model any tiny unobserved interactions.

Finally, the following could be derived by applying this to the LTC formula:

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + S(t) + \sigma\big(x(t)\big)B$$

$$Replace\ S(t)\ with\ the\ nonlinearity\ proposed,$$

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} + f(x(t), I(t), t, \theta)(A - x(t)) + \sigma\big(x(t)\big)B$$

$$Expand\ out\ the\ equation,$$

$$\frac{dx(t)}{dt} = \frac{-x(t)}{\tau} - f(x(t), I(t), t, \theta)x(t) + \sigma\big(x(t)\big)B + f(x(t), I(t), t, \theta)A$$

$$Lastly, refactor\ the\ equation\ into\ the\ format\ of\ the\ original\ LTC$$

$$\boxed{\frac{dx(t)}{dt} = -\left[\frac{1}{\tau} + f(x(t), I(t), t, \theta) - \sigma B\right]x(t) + f(x(t), I(t), t, \theta)A}$$

Algorithm forward propagation by SDE solvers

Hasani et al. (2020) determined that their LTC architecture that uses a linear system of ODEs was 'stiff equations'. They also found that regular Runge-Kutta was not suitable for solving LTCs; therefore, they designed a custom ODE solver by combining both implicit and explicit Euler methods.

As this system uses SDEs, SDE solvers must be used. As Hasani et al. (2020) determined, the architecture is a system of stiff equations. Therefore, as Press et al. (2007) decided, researchers must use an implicit solver to ensure stability. Additionally, researchers can combine an explicit solver to achieve further stability. Therefore, the author will use an SDE solver, which is implicit, and if time permits, create a further enhanced custom SDE solver by fusing an explicit solver within.

Based on the author's research, the SDE equivalent for ODE Euler methods is the Euler-Maruyama method; this is the recommended solver as it can handle all forms of noise (Li et al., 2020). The author will therefore use the implicit Euler-Maruyama solver and, if time permits, combine the explicit solver within.

How to train the network?

Training these networks has a trade-off between accuracy and memory. Chen et al. (2019) promoted the use of the adjoint sensitivity method to perform reverse-mode AD, which is more memory efficient. Hasani et al. (2020) mentioned that this method introduced more numerical errors and opted to use the traditional BPTT approach, which is more accurate but consumes more memory. Although there exists a technique of adjoints specifically for SDEs, they cannot be used, as determined by Tzen and Raginsky (2019), and hence requires a custom-built backpropagation rule.

For this research, the author will opt for the approach by Hasani et al. (2020) to give more precision and as the author is time constrained to implement a custom backpropagation algorithm. Researchers must investigate reverse-mode AD in the future as it is the recommended approach when memory efficiency is more important. It is also worth noting that using the BPTT approach carries added benefits, such as being able to be used as an RNN layer alongside the popular optimization algorithms that are very familiar (ex: Adam, SGD) (Hasani et al., 2020).

4.4. UI design

The author had decided to implement a web application for the supplementary application being built due to convenience. The wireframes designed to be of use are available at **APPENDIX II**.

4.5. System process flow chart

A summarized system flow chart that end-users will follow is presented in the diagram below.

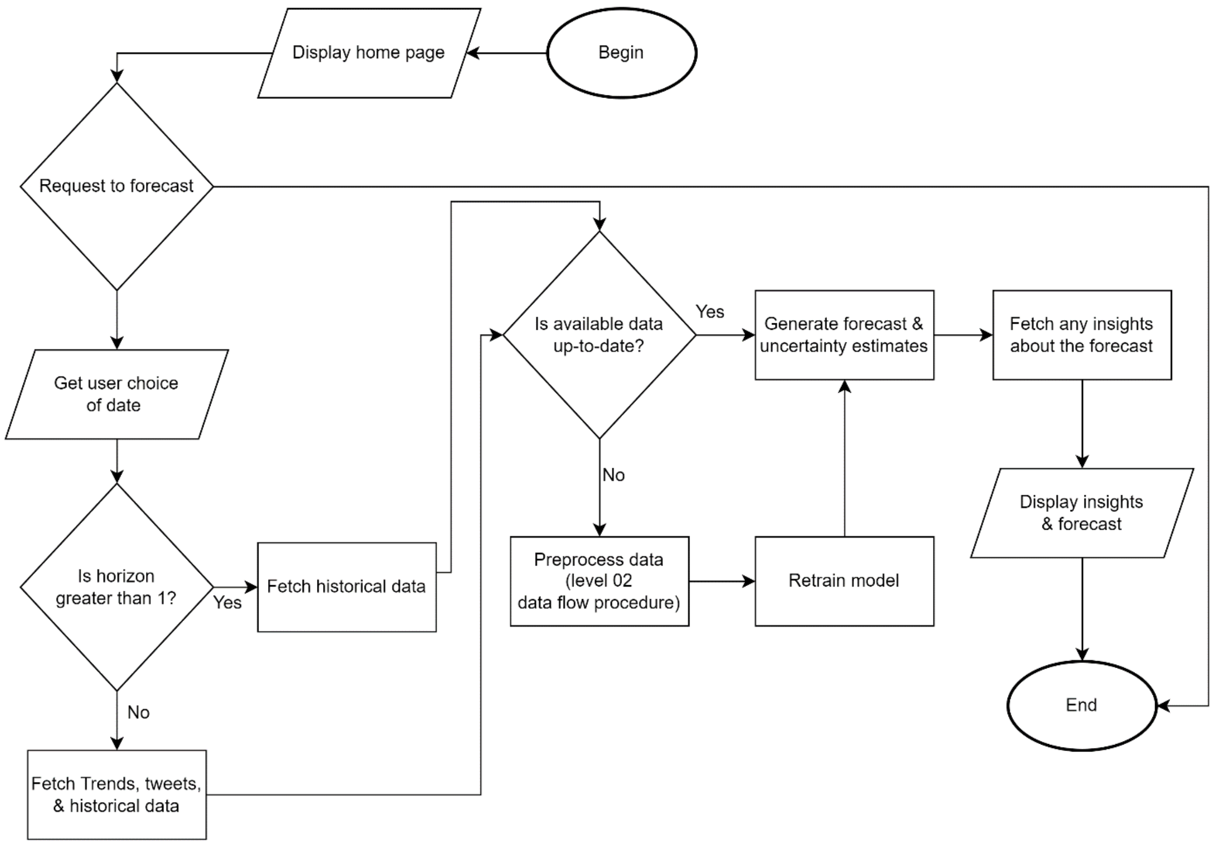

Figure 4: System process flow chart (*Self-Composed*)

REFERENCES

Hasani, R. et al. (2020). Liquid Time-constant Networks. Available from https://doi.org/10.48550/arXiv.2006.04439 [Accessed 25 September 2022].

Duvenaud, D (2021). Directions in ML: Latent Stochastic Differential Equations: An Unexplored Model Class. *YouTube*. Available from https://www.youtube.com/watch?v=6iEjF08xgBg. [Accessed on 30 Sep. 2022].

Pontryagin, L.S., Boltyanskii, V.G., et al. (1962) The Mathematical Theory of Optimal Processes. Interscience Publishers John Wiley & Sons, Inc., New York-London, Translated from the Russian by K.N., Trirogoff.

Tzen, B. and Raginsky, M. (2019). Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit. Available from http://arxiv.org/abs/1905.09883 [Accessed 2 December 2022].

Chen, R.T.Q. et al. (2019). Neural Ordinary Differential Equations. Available from https://doi.org/10.48550/arXiv.1806.07366 [Accessed 25 September 2022].

Press, W.H. (ed.). (2007). *Numerical recipes: the art of scientific computing*, 3rd ed. Cambridge, UK ; New York: Cambridge University Press.

Lapicque, L. 1907. Recherches quantitatives sur l'excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et de Pathologie Generalej* 9: 620–635.

Li, X. et al. (2020). Scalable Gradients for Stochastic Differential Equations. Available from http://arxiv.org/abs/2001.01328 [Accessed 18 January 2023].

APPENDIX I – Algorithm Intuition

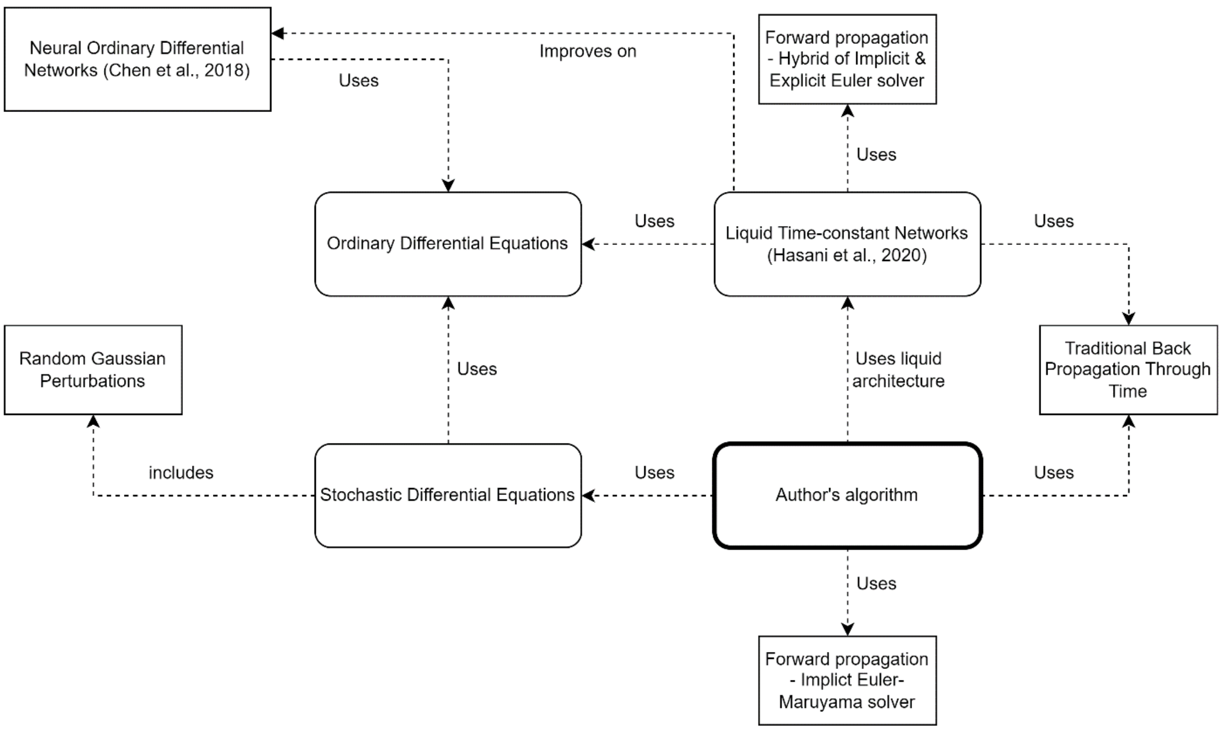

Figure 5: Algorithm intuition (*Self-Composed*)

APPENDIX II – UI Wireframes

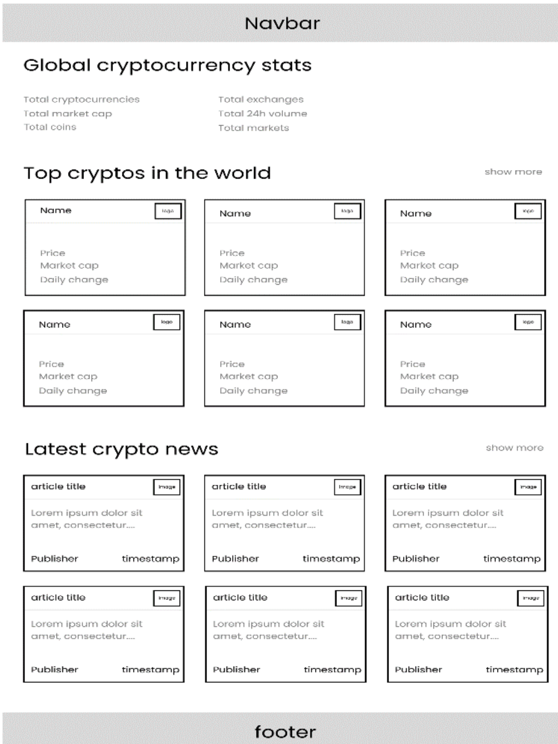

Figure 6: UI wireframes – Home (*Self-Composed*)

Figure 7: UI wireframes – News (*Self-Composed*)

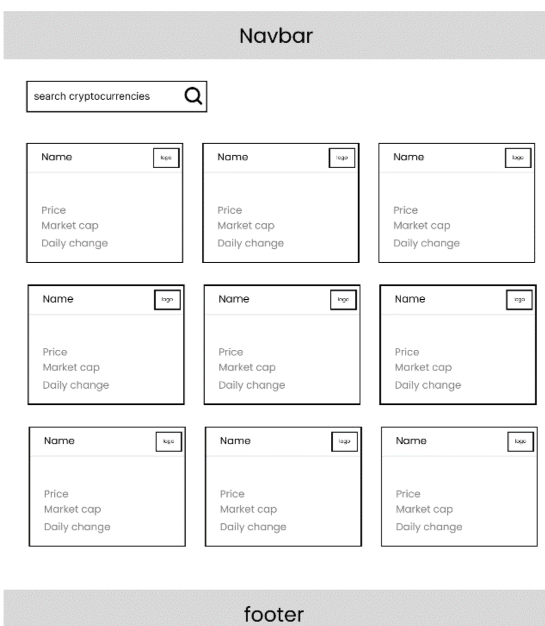

Figure 8: UI wireframes – Cryptocurrencies (*Self-Composed*)

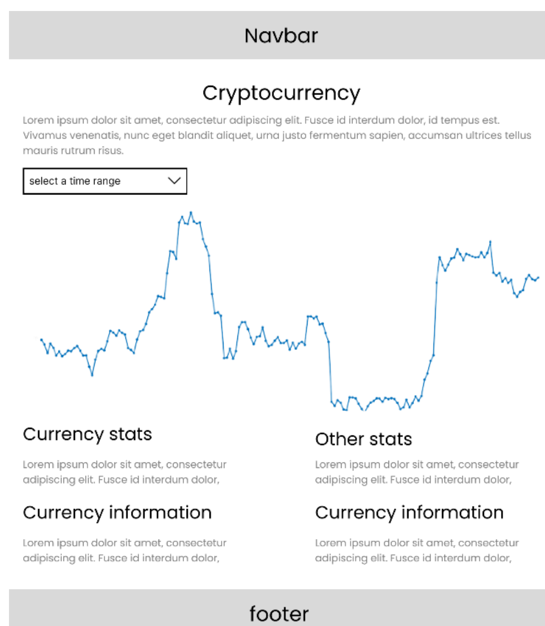

Figure 9: UI wireframes – Cryptocurrency (*Self-Composed*)

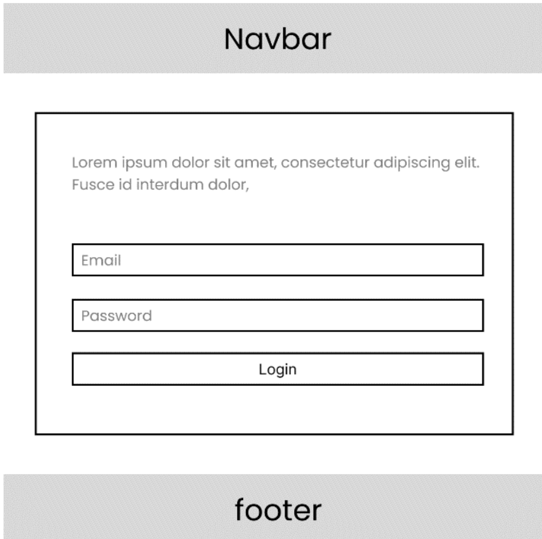

Figure 10: UI wireframes - Admin login (*Self-Composed*)

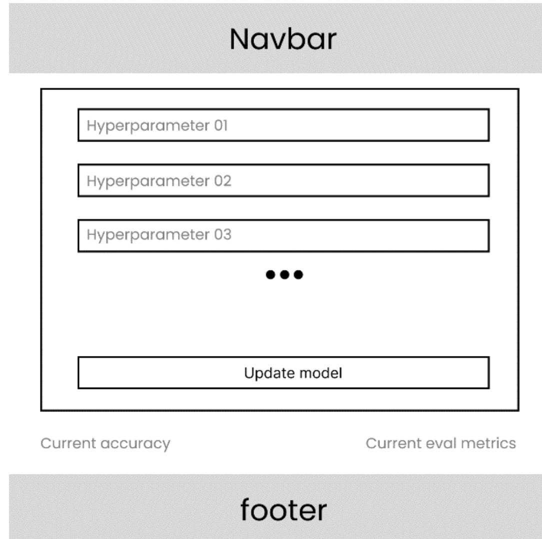

Figure 11: UI wireframes – Admin model configuration (*Self-Composed*)

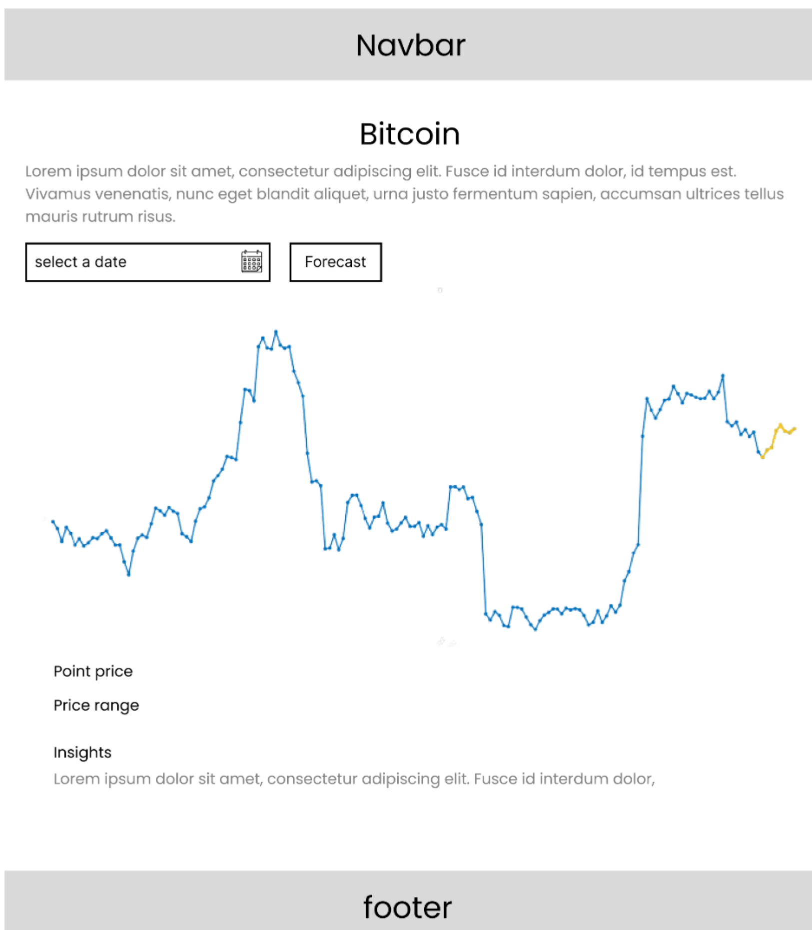

Figure 12: UI wireframes - Forecast (*Self-Composed*)