

SmartCost: Business Performance Prediction System for Micro-SMEs

INDIVIDUAL THESIS

K A C H Kodithuwakku

(IT22544872)

Individual component: Sales, Cost and Profit Forecasting

Group ID: R26-IT-026

Report submitted in partial fulfilment of the requirements for the

B.Sc. (Hons) Degree in Information Technology

Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

November 2026

DECLARATION AND SUPERVISOR CERTIFICATION

Candidate declaration. I declare that this thesis is my own work and does not incorporate, without acknowledgement, material previously submitted for a degree or diploma at any university or institute of higher learning. To the best of my knowledge, it contains no material previously published or written by another person except where acknowledgement is made in the text. I grant the Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute this thesis, in whole or in part, in print, electronic, or other media, while retaining the right to use the content in future work.

Academic integrity and AI disclosure. I have disclosed all use of generative AI and other automated tools in accordance with the module policy. I remain responsible for the accuracy, originality, citations, code, data, analyses, and conclusions in this thesis.

Student name: K A C H Kodithuwakku

Student ID: IT22544872

Signature: 

Date: 4 October 2026

Supervisor certification. The above candidate carried out this research for the bachelor's degree thesis under my supervision.

Role and name	Signature	Date
Supervisor: Deemantha Siriwardana		8 October 2026
Co-supervisor: Ayesha Wijesooriya		8 October 2026

ABSTRACT

Small restaurants record every sale in their point-of-sale (POS) systems, yet most of these systems report only the past, and owners plan purchases, staff and cash by intuition. Existing forecasting research mostly assumes long histories and expert users. This thesis designs, implements and evaluates a forecasting component for SmartCost POS, a multi-tenant POS for Sri Lankan businesses, that predicts monthly sales, costs and profit for each branch over 1, 3, 6 and 12 months with 95% prediction ranges. A data-size-aware model ladder selects a moving average, damped Holt or damped Holt-Winters model according to the months of history available; models are fitted in a Python analysis service and shown in the mobile app with their measured accuracy and a plain-language data-quality message. A rolling-origin evaluation on 30 simulated restaurant branches, a noisier scenario and system-produced data (267,468 scored forecasts) compared the ladder with five baselines. With 24 or more months, Holt-Winters reduced sales error one to three months ahead from 7.4% (seasonal naive) to 4.3% MAPE and was more accurate in 81.9% of paired cases ($p < 0.001$). From 12 months of history onwards, the ladder's 3-month totals were within 4.4% for sales and 3.5% for costs. With shorter histories, simpler rules were more accurate; a revised ladder based on them lowered one-month sales error from 7.5% to 5.4%. The ranges covered 81.4% of actuals one month ahead and 99.1% at twelve months, showing they are too narrow in the short term and too wide in the long term. The evaluation also showed that unrecorded months drive forecasts to zero. All 18 automated tests passed. The work contributes an evaluated forecasting design for young business histories and a reusable analysis architecture adopted by three other components.

Keywords: sales forecasting, exponential smoothing, point-of-sale systems, small and medium enterprises, prediction intervals

ACKNOWLEDGEMENTS

I thank my supervisor, Deemantha Siriwardana, and my co-supervisor, Ayesha Wijesooriya, for their guidance throughout this research. I am grateful to my group members, N.K.M. Perera, M.P.J.R. Dasanayaka and D.D.M. Jayasingha, with whom the SmartCost POS platform was built and whose components share the analysis service described in this thesis. I also thank the three businesses that use SmartCost in their daily operation, and the Sri Lanka Institute of Information Technology for the facilities provided.

TABLE OF CONTENTS

1 INTRODUCTION	1
1.1 Overall project context	1
1.2 Individual component context	2
1.3 Problem statement	2
1.4 Research gap	3
1.5 Aim and objectives	3
1.6 Research questions or hypotheses	4
1.7 Scope, limitations and assumptions	4
1.8 Individual contribution	4
1.9 Thesis structure	5
2 LITERATURE REVIEW AND RELATED WORK	6
2.1 Review method and scope	6
2.2 Domain and theoretical foundations	7
2.2.1 Forecasting in restaurants and small businesses	7
2.2.2 Exponential smoothing	7
2.2.3 Alternative model families	8
2.2.4 How much history is needed	9
2.2.5 Evaluating forecasts	9
2.2.6 Communicating uncertainty to non-experts	9
2.3 Related research	10
2.4 Existing systems, products and techniques	11
2.5 Gap synthesis and conceptual framework	12
3 RESEARCH METHODOLOGY	14
3.1 Research design	14
3.2 Data, participants or study materials	14
3.3 Experimental or study procedure	15
3.4 Measures and evaluation metrics	16
3.5 Data analysis	17
3.6 Ethics, privacy, security and responsible AI	18
3.7 Reproducibility and research artefacts	18
4 PRODUCT AND SOLUTION DESIGN	20

4.1 Requirements	20
4.2 Component architecture within the complete solution	21
4.3 Design alternatives and decisions	23
4.3.1 Exponential smoothing rather than ARIMA or machine learning	24
4.3.2 A model ladder chosen by the amount of history	24
4.3.3 Excluding the month in progress	25
4.4 Detailed design	25
4.4.1 Data sources and monthly aggregation	25
4.4.2 Request flow	26
4.4.3 Model selection and forecasting algorithm	26
4.4.4 Interface design	28
4.4.5 User interface design	29
4.5 Standards and professional practice	32
5 IMPLEMENTATION, INTEGRATION AND TESTING	33
5.1 Development environment and technology stack	33
5.2 Component implementation	33
5.2.1 Monthly data preparation	33
5.2.2 Orchestration	34
5.2.3 Bridge to the analysis service	34
5.2.4 Forecasting engine	35
5.2.5 Mobile user interface	35
5.2.6 Synthetic data generator	35
5.3 Integration with the other components	36
5.4 Verification and software testing	39
5.5 Deployment and operation	41
6 RESULTS AND VALIDATION	43
6.1 Research results	43
6.1.1 Accuracy by length of history (RQ1)	43
6.1.2 Accuracy by forecast horizon on common folds (RQ1)	45
6.1.3 Accuracy of the totals shown to the owner (RQ1)	45
6.1.4 Coverage of the prediction ranges (RQ2)	46
6.1.5 Worked example on system-produced data	47
6.1.6 Robustness to unrecorded months (dataset S4)	47

6.2 Product and technical results	48
6.3 User or stakeholder validation	49
6.4 Integration results	49
6.5 Summary of findings	50
7 DISCUSSION	52
7.1 Interpretation of findings	52
7.2 Comparison with literature baselines and existing solutions	53
7.3 Research contribution	54
7.4 Product contribution	54
7.5 Limitations and threats to validity	54
7.6 Implications	55
8 COMMERCIALISATION AND SUSTAINABILITY	56
8.1 Value proposition and target users	56
8.2 Market and competitive position	57
8.3 Business or sustainability model	57
8.4 Scalability, intellectual property and compliance	58
8.4.1 Scalability	58
8.4.2 Intellectual property	58
8.4.3 Legal and regulatory compliance	59
8.5 Contribution to the Sustainable Development Goals	59
9 CONCLUSION AND FUTURE DIRECTIONS	61
9.1 Conclusion	61
9.2 Achievement of objectives	61
9.3 Individual research and product contributions	62
9.4 Future directions	62
REFERENCES	63
GLOSSARY	66
APPENDICES	68
Appendix A Research instruments or extended results	68
A.1 Extended forecasting results	68
A.2 User study instruments	70
Appendix B Academic integrity and AI use disclosure	71

LIST OF FIGURES

Figure 4.1: Architecture of the forecasting component within SmartCost POS	22
Figure 4.2: Sequence of interactions for one forecast request	26
Figure 4.3: Model selection ladder and fallback chain	27
Figure 4.4: Header and horizon selector of the forecast section	30
Figure 4.5: Forecast section for the development branch with 36 months of seeded data: (a) prediction cards for the next three months; (b) month in progress and History vs forecast chart	31
Figure 4.6: Month by month breakdown of the three-month forecast	31
Figure 4.7: Model accuracy panel with backtest errors, history length and model name	32
Figure 5.1: Monthly sales aggregation in forecastDataService.js (simplified)	34
Figure 5.2: Model selection and prediction ranges in forecasting.py (simplified)	35
Figure 5.3: Analytics screen of the mobile app, with the forecast section below the period summary	38
Figure 6.1: Forecast error one to three months ahead by months of history, dataset S1 (the revised ladder coincides with the 3-month average below 12 months, the seasonal naive at 12–23 months and the production ladder from 24 months)	44
Figure 6.2: Share of actual values inside the 95% prediction range by horizon, dataset S1	46
Figure 6.3: Holdout forecast for March to July 2026 on the development branch history (S3)	47
Figure 6.4: Effect of two unrecorded months on the forecast (S4) compared with the history trimmed to July 2026	48
Figure 6.5: Forecast section with no network connection (TC-S4)	50

LIST OF TABLES

Table 1.1: Research components of the SmartCost POS project	1
Table 2.1: Themes of the literature review	6
Table 2.2: Comparison of related studies	10
Table 2.3: Forecasting features of existing POS products	11
Table 2.4: Research gaps and how this thesis addresses them	12
Table 3.1: Research questions, methods and evidence	14
Table 3.2: Datasets used in the evaluation	14
Table 3.3: Methods compared	16
Table 3.4: Measures and evaluation metrics	17
Table 3.5: Commands to reproduce the results (run in analysis-backend)	18
Table 4.1: Functional requirements of the forecasting component	20
Table 4.2: Non-functional requirements of the forecasting component	21
Table 4.3: Design decisions and alternatives considered	23
Table 4.4: Data fields used by the forecasting component	25
Table 4.5: API endpoints of the forecasting component	28
Table 4.6: Data-quality levels reported to the user	29
Table 5.1: Technology stack of the forecasting component	33
Table 5.2: Parameters of the synthetic restaurant data generator	36
Table 5.3: Test cases for the forecasting component	39
Table 5.4: Deployment steps	41
Table 6.1: Sales MAPE (%) one to three months ahead by months of history, dataset S1	43
Table 6.2: Paired comparison of absolute errors one month ahead (Wilcoxon signed-rank test)	44
Table 6.3: Sales MAPE (%) by horizon on common folds (origins of 12 months or more)	45
Table 6.4: Error of the 3- and 12-month totals (MAPE of the total, %), common folds	46
Table 6.5: Coverage (%) and mean width (% of actual) of the 95% ranges, dataset S1	46
Table 6.6: Summary of findings against the research questions and objectives	50
Table 7.1: Threats to validity	54

Table 8.1: Price and forecasting scope of competing POS products (vendor pages, 4 October 2026)	57
Table 8.2: Licences of the main libraries used by the component	59
Table 8.3: Contribution to the Sustainable Development Goals	59
Table 9.1: Achievement of objectives	61
Table A.1: Costs MAPE (%) one to three months ahead by months of history, dataset S1	68
Table A.2: Sales MAPE (%) one to three months ahead by months of history, dataset S2	68
Table A.3: MASE by horizon on common folds, dataset S1	69
Table A.4: Costs MAPE (%) by horizon on common folds	69
Table A.5: Paired comparison of absolute errors three months ahead (Wilcoxon signed-rank test)	69
Table B.1: Use of generative AI tools	71

LIST OF ABBREVIATIONS

Abbreviation	Meaning
AI	Artificial intelligence
API	Application programming interface
ARIMA	Autoregressive integrated moving average
ASGI	Asynchronous Server Gateway Interface
ETS	Error, trend, seasonality (exponential smoothing state space models)
FR	Functional requirement
GAM	Generalised additive model
GPU	Graphics processing unit
HTTP	Hypertext Transfer Protocol
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
JWT	JSON Web Token
LKR	Sri Lankan rupee
LSTM	Long short-term memory
MA	Moving average
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MASE	Mean absolute scaled error
ML	Machine learning
MLP	Multilayer perceptron
NFR	Non-functional requirement
OWASP	Open Worldwide Application Security Project
PEP	Python Enhancement Proposal
POS	Point of sale
QR	Quick response (code)
REST	Representational state transfer
RF	Random forest

Abbreviation	Meaning
RFC	Request for Comments
RMSE	Root mean squared error
RQ	Research question
SARIMA	Seasonal ARIMA
SD	Standard deviation
SDG	Sustainable Development Goal
SES	Simple exponential smoothing
SLIIT	Sri Lanka Institute of Information Technology
SME	Small and medium-sized enterprise
SMS	Short Message Service
SUS	System Usability Scale
TBATS	Exponential smoothing with trigonometric seasonality, Box-Cox transformation, ARMA errors, trend and seasonal components
TFT	Temporal fusion transformer
URL	Uniform resource locator
USD	United States dollar
UTC	Coordinated Universal Time

1 INTRODUCTION

Point-of-sale (POS) systems record every sale a restaurant makes, yet most of them only report what has already happened. This chapter introduces the SmartCost POS research project, the sales and cost forecasting component that is the subject of this thesis, the problem and research gap it addresses, and the objectives and research questions that structure the rest of the work.

1.1 Overall project context

SmartCost POS is a multi-tenant point-of-sale platform for small and medium-sized shops and restaurants in Sri Lanka, developed by research group R26-IT-026. It consists of a Node.js and Express backend with a MongoDB database, a Next.js web dashboard, an Expo/React Native mobile POS application and, for the research components, a Python FastAPI analysis service. The platform supports multiple shops and branches, product and category management, carts and checkout, table and kitchen management for restaurants, customer QR ordering, cost and expense recording, subscriptions and SMS notifications.

Beyond its operational features, the project investigates how the data that a small business already records in its POS can be turned into decision support. Each of the four group members owns one research component, listed in Table 1.1. All four read the same transaction data and share the same analysis service.

Table 1.1: Research components of the SmartCost POS project

No.	Component	Question it answers for the owner
1	Sales, cost and profit forecasting (this thesis)	How much will we sell and spend in the coming months?
2	Customer behaviour analytics	When do customers come, what do they buy, and which customer segments exist?
3	Product demand forecasting	How much of each product will be needed?
4	Recommendation system	Which item should be suggested to a customer at checkout?

1.2 Individual component context

This thesis covers the first component: a forecasting service that predicts monthly sales, costs and profit for a shop branch over the next 1, 3, 6 and 12 months. It reads completed orders from the History collection and expenses from the CostExpense collection, builds monthly series, fits forecasting models in the Python analysis service, and presents the result in the Analytics screen of the mobile app with a prediction range, the measured accuracy of the model and a plain-language message about how much history the forecast is based on.

The component was the first of the four to be built. Its split between data preparation in Node.js and modelling in Python became the architecture that the other three components reused.

1.3 Problem statement

Small and medium restaurant owners plan ingredient purchases, staffing and cash flow largely by intuition. Their POS systems hold months or years of transaction data, but the reports of most POS products describe only past periods (Section 2.4), even though demand forecasting is recognised as an important input to restaurant revenue management [1]. Restaurant demand is strongly seasonal: in Sri Lanka it rises around the Sinhala and Tamil New Year in April and the December holiday season, so intuition based on the last few weeks can lead to stock-outs in peak months and overspending in slow ones.

Forecasting methods that handle trend and seasonality are well established [2], but applying them in a small business raises three practical difficulties. First, a young business has only a short history, and seasonal models need at least two years of monthly data. Second, the owner is not a statistician and cannot judge which model to use or how far to trust its output. Third, the forecast must work automatically inside the POS the owner already uses, for every branch, without an analyst.

Problem: small restaurants using a POS have no automatic, trustworthy forecast of their future sales, costs and profit that works with the short and growing histories they actually have and that a non-expert owner can interpret.

1.4 Research gap

The literature review in Chapter 2 identifies four gaps that this work addresses:

1. **Short and growing histories.** Forecasting studies and competitions mostly evaluate methods on long series [3], [4]. Little work examines how a forecasting system for a new business should behave as its history grows from a few months to several years, and which model is appropriate at each stage.
2. **Uncertainty for non-experts.** Prediction intervals are standard in forecasting theory [2], but their calibration on short business series, and their presentation to owners without statistical training, are rarely evaluated.
3. **Integration into everyday tools.** Restaurant forecasting studies typically report model accuracy offline [1], [5], [6]; few describe a forecast embedded in a working multi-tenant POS with authentication, branch isolation and a mobile interface.
4. **Local context.** Commercial POS products used in Sri Lanka offer historical reports but, as Section 2.4 shows, little or no forward-looking sales, cost and profit forecasting with uncertainty.

1.5 Aim and objectives

Aim: to design, implement and evaluate an automatic forecasting component for SmartCost POS that predicts monthly sales, costs and profit for a restaurant branch, adapts to the amount of history available, and communicates the forecast and its uncertainty to a non-expert owner.

The aim is broken down into five objectives:

1. **O1.** Build a data pipeline that turns POS orders and expenses into evenly spaced monthly series per branch, excluding the month in progress.
2. **O2.** Design a model ladder that selects a forecasting model according to the number of complete months of history.
3. **O3.** Produce 1, 3, 6 and 12-month forecasts of sales, costs and profit with 95% prediction ranges, accuracy figures and a data-quality message.
4. **O4.** Evaluate forecast accuracy and range coverage by rolling-origin backtesting against simple baseline methods, for different lengths of history.

5. **O5.** Integrate the component into the SmartCost backend and mobile app, verify it with automated tests, and validate its usefulness with restaurant owners.

1.6 Research questions or hypotheses

The evaluation is structured around three research questions:

- **RQ1.** How accurate is the data-size-aware model ladder, compared with simple baseline methods, at different lengths of history and forecast horizons?
- **RQ2.** How well do the 95% prediction ranges produced by the component cover the actual values?
- **RQ3.** Do restaurant owners find the forecast understandable and useful for planning?

RQ1 and RQ2 are answered quantitatively in Section 6.1. RQ3 is to be answered by the user study designed in Section 3.3; Section 6.3 reports its status.

1.7 Scope, limitations and assumptions

- **Granularity:** monthly totals only; daily and weekly forecasting is out of scope.
- **Unit of analysis:** one shop branch at a time; sales and costs are forecast as separate series and profit is derived from them.
- **Inputs:** only the branch's own history; external factors such as weather, holidays, promotions and prices are not modelled.
- **Data:** the quantitative evaluation uses simulated restaurant data generated with the project's seed generator and the branch history stored in the development system; SmartCost has been in live use at three businesses since September 2026 (Section 5.5), but their history is still too short to evaluate.
- **Assumption:** a branch records all its sales through SmartCost and enters its expenses with a purchase date, so the recorded totals represent the business.

1.8 Individual contribution

The author designed and implemented the complete forecasting component, including:

- the monthly aggregation service (forecastDataService.js), controller and route in the Node.js backend;
- the forecasting engine (forecasting.py), its FastAPI endpoint and Pydantic schemas in the Python analysis service, which was created for this component;
- the shared Node.js client for the analysis service, later reused by the other three components;
- the Forecast section of the mobile app;
- the synthetic restaurant data generator, the automated test suites and the evaluation scripts used in this thesis.

The component reuses the platform's authentication middleware, the History and CostExpense data models and the Analytics screen, which were built as part of the shared SmartCost platform.

1.9 Thesis structure

Chapter 2 reviews the forecasting literature and existing POS products and derives the research gap. Chapter 3 describes the research methodology and evaluation design. Chapter 4 presents the requirements and design of the component, and Chapter 5 its implementation, integration and testing. Chapter 6 reports the results, which Chapter 7 interprets against the literature, together with the limitations. Chapter 8 discusses commercialisation and sustainability, and Chapter 9 concludes and proposes future work.

2 LITERATURE REVIEW AND RELATED WORK

This chapter reviews the knowledge on which the forecasting component builds. It explains how the review was carried out, summarises the forecasting methods and evaluation practices that are relevant to short monthly business series, examines recent research on restaurant and food demand forecasting and on communicating uncertainty, surveys the forecasting features of existing POS products, and derives the research gap.

2.1 Review method and scope

The review is a structured narrative review rather than a systematic review. Sources were identified by keyword searches in Google Scholar and the publisher platforms of the *International Journal of Forecasting*, Springer and PLoS, using combinations of the terms *restaurant*, *food service*, *sales forecasting*, *demand forecasting*, *exponential smoothing*, *short time series*, *small business*, *prediction interval* and *uncertainty communication*, and by following the references of the review papers found. Recent studies were restricted to 2019–2026. Foundational works were included regardless of age when the component uses them directly.

A source was included if it (a) proposed or evaluated a method usable on a single short monthly series, (b) forecast restaurant, food or retail sales, (c) addressed how forecasts should be evaluated, or (d) studied how people understand numerical uncertainty. The bibliographic details of every source were checked against Crossref, the publisher or the author's page. Commercial products were reviewed from their public documentation and pricing pages, accessed on 4 October 2026. Table 2.1 summarises the themes.

Table 2.1: Themes of the literature review

Theme	Question for this thesis	Main sources
Forecasting methods	Which methods suit a branch with 3 to 36 monthly observations?	[3], [4], [7]–[12]
Restaurant and retail demand	What has been done for restaurants, with what data and how was it evaluated?	[1], [5], [6], [13]

Theme	Question for this thesis	Main sources
Forecast evaluation	How should accuracy and prediction ranges be measured?	[14], [15]
Uncertainty and acceptance	Can non-experts read a forecast with a range, and how is usefulness measured?	[16]–[18]
Existing products	What do commercial POS systems already offer?	Vendor documentation (Section 2.4)

2.2 Domain and theoretical foundations

2.2.1 Forecasting in restaurants and small businesses

Demand forecasting is one of the main inputs to restaurant revenue management, staffing and purchasing [1]. Lasek et al. grouped the methods used for restaurant sales and guest counts into seven categories, from naive and moving-average rules through exponential smoothing and regression to neural networks and hybrids [1]. The component in this thesis works at the aggregate level: monthly totals of sales and costs per branch, used for purchasing, staffing and cash-flow planning.

A small business differs from the settings of most forecasting research in two ways. Its history is short: a branch that opened last year has at most twelve monthly observations. And it has no analyst, so model choice, parameter estimation and the judgement of reliability must all be automatic.

2.2.2 Exponential smoothing

Exponential smoothing forecasts a series as a weighted average of past observations, with weights that decrease exponentially with age. Holt extended the method with a trend component [7], and Winters added a seasonal component in a paper explicitly about forecasting sales [8]; the resulting Holt-Winters method is the model used by the component once two years of data exist. Gardner and McKenzie introduced the damped trend, which makes the trend fade with the horizon, and showed on the 1,001 series of the M-competition that it improved accuracy, particularly at long lead times [9]. In a later review of the field, Gardner concluded that research on selecting among exponential smoothing methods was inconclusive and that it was still difficult

to beat applying a damped trend to every series [10]. This is the main justification for damping both the Holt and the Holt-Winters rungs of the ladder (Section 4.3).

Every exponential smoothing method also corresponds to a state space model (the ETS family), which provides a likelihood, information criteria for automatic model selection and model-based prediction intervals [2]. The component in this thesis takes a simpler route: it chooses the model by a rule based on the length of the history rather than by an information criterion, and it computes ranges from the residual spread. For a 95% interval, the textbook formula is the forecast $\pm 1.96 \hat{\sigma}_h$, where $\hat{\sigma}_h$ is the standard deviation of the h-step forecast distribution; for the naive method $\hat{\sigma}_h = \hat{\sigma} \sqrt{h}$ [2]. The component's range is therefore exact for a random walk but only an approximation for damped and seasonal models, a point examined empirically in Chapter 6.

2.2.3 Alternative model families

ARIMA. Box-Jenkins ARIMA models describe a series through its autocorrelation after differencing [2]. Their identification (choosing the orders of differencing, autoregression and moving average, and their seasonal counterparts) needs considerably more observations than a young branch has.

Prophet. Taylor and Letham's Prophet fits a decomposable regression with trend, seasonality and holiday terms, designed for an "analyst-in-the-loop" who adjusts interpretable parameters [11]. It is aimed at daily business data and assumes an analyst, neither of which applies to monthly forecasts for owners.

Machine learning. Recurrent networks such as the long short-term memory (LSTM) network and gradient-boosted trees can learn complex patterns, but they need large amounts of data. On 1,045 monthly series from the M3 competition, Makridakis et al. found that machine-learning methods were less accurate than simple statistical methods at every horizon and far more expensive to compute [12]. In the M4 competition, twelve of the seventeen most accurate methods were combinations of mostly statistical methods, and none of the pure machine-learning entries beat the combination benchmark [3]. In M5, machine learning won, but on 42,840 related Walmart series where models could learn across series and use external variables [4].

A single restaurant branch with a few dozen monthly totals has none of these advantages.

2.2.4 How much history is needed

A seasonal model must see each month of the year at least twice before it can distinguish a seasonal pattern from noise and trend, which is why the component starts Holt-Winters at 24 months. Hyndman and Kostenko showed that such theoretical minimums do not guarantee good seasonal estimates, that the amount of data actually needed depends on how noisy the series is, and that there is no simple rule [19]. The thresholds of a data-size ladder therefore have to be justified empirically, which is what RQ1 does.

2.2.5 Evaluating forecasts

Tashman recommended out-of-sample evaluation with a rolling forecast origin, re-estimating the model at each origin, so that a method is judged on many forecasts rather than a single holdout period [14]; the same procedure is called time-series cross-validation in the textbook [2]. Hyndman and Koehler showed that common measures such as MAPE become undefined or misleading when actual values are close to zero, and proposed the mean absolute scaled error (MASE) [15]. For prediction intervals, coverage alone is not enough, because a very wide interval always covers; an evaluation must also consider how wide the interval is [2]. The M4 competition evaluated prediction intervals alongside point forecasts for this reason [3].

2.2.6 Communicating uncertainty to non-experts

In five experiments with 5,780 participants, including a field experiment on the BBC News website, van der Bles et al. found that presenting a numerical range reduced trust in the number only slightly, while verbal expressions of uncertainty reduced it more [16]. This supports showing owners a numerical range rather than hiding uncertainty or describing it in words only. Whether owners find such a forecast useful can be measured with the perceived-usefulness and ease-of-use constructs of the technology acceptance model [17] and with the System Usability Scale [18].

2.3 Related research

Table 2.2 compares recent studies that forecast restaurant and food demand.

Schmidt et al. compared linear models, tree ensembles and recurrent and transformer networks on the sales of one mid-sized restaurant and obtained a best symmetric MAPE of about 19.6% one day ahead and 19.5% one week ahead [5]. Posch et al. forecast daily sales of menu items in a restaurant and a staff canteen from POS data covering 20 months, and found that a Bayesian generalised additive model with a negative binomial likelihood outperformed Prophet, SARIMA, ETS and TBATS; unusually, they also evaluated the prediction intervals, with coverage, width and the interval score [6]. Hossain and Parvin, working with 15 months of data from a restaurant in Dhaka, found that a multilayer perceptron was most accurate but still had a MAPE of about 46%, and that simple exponential smoothing beat XGBoost; they note that the short history constrained the machine-learning models [13].

Table 2.2: Comparison of related studies

Study	Data	Granularity and history	Methods	Ranges evaluated	Gap relative to this thesis
Lasek et al. 2016 [1]	Survey of restaurant studies	Mostly daily or hourly	Seven method categories	Not a focus	No evaluation; no small-business or short-history focus
Schmidt et al. 2022 [5]	One restaurant	Daily; several years	Linear, ensembles, LSTM, TFT	No	No monthly or short-history analysis; not in a POS
Posch et al. 2022 [6]	Restaurant and canteen POS data	Daily items; 20 months	Bayesian GAM vs ETS, SARIMA, Prophet	Yes	No costs or profit; no model choice by history length
Hossain and Parvin 2025 [13]	One Dhaka restaurant, five items	Daily items; 15 months	SES, Croston, RF, XGBoost, MLP	No	Single holdout; no ranges; not in a POS
This thesis	Simulated branches and system data	Monthly totals; 3–36 months	Ladder: MA, damped Holt, damped Holt-Winters	Yes (coverage, width)	—

Three observations follow from Table 2.2. First, almost all studies use daily or item-level data, where many observations are available; none studies how a forecasting method should change as a business accumulates monthly history. Second, only one study evaluates prediction intervals, and none examines how intervals are presented

to an owner. Third, none forecasts costs or profit, and none describes a forecast running inside a multi-tenant POS. The study closest in data size, that of Hossain and Parvin, shows how difficult accurate forecasting is with 15 months of data.

2.4 Existing systems, products and techniques

Table 2.3 summarises the forecasting features documented by widely used international POS products and by two POS products marketed in Sri Lanka. The review is based on public vendor documentation accessed on 4 October 2026; a feature that is not documented may still exist, and the table records only what was found.

Table 2.3: Forecasting features of existing POS products

Product	Sales forecast	Cost or profit forecast	Range shown	History required
Square for Restaurants	Not documented	Labour-cost projection from scheduled shifts [20]	Not documented	Not documented
Toast	Daily net sales and order volume, 7 days ahead (up to 28 with a paid tier) [21]	Projected sales used in staff scheduling [22]	Yes: projected range and confidence banner [21]	At least 30 days of history [21]
Lightspeed Restaurant	Benchmarks and Trends reports on Essential and Premium plans [23]; no sales forecast documented	No profit forecast documented	Not documented	Not documented
possystem.lk (Sri Lanka)	"AI sales forecasting" of daily sales and busy hours; method not described [24]	Historical margins only	Not documented	Not documented
TILLMAX (Sri Lanka)	Not documented [25]	Not documented	—	—
SmartCost (this thesis)	Monthly, 1–12 months ahead	Monthly costs and derived profit	Yes: 95% range, accuracy and data-quality message	From 3 months; model chosen by history length

Toast is the only product found that documents both a forecast and a range. Its forecasts, however, cover daily sales a few weeks ahead for staffing; it does not forecast costs or profit, and it does not publish how its ranges are computed or how well they are calibrated. The documentation of Sling, the scheduling tool integrated with Toast, says that a discrepancy of 30–40% between projected and actual sales is

normal [22]. The Sri Lankan products found offer historical reporting, and one claims AI-based daily forecasting without describing it.

On the technique side, open-source libraries make the methods of Section 2.2 readily available: statsmodels implements exponential smoothing and ARIMA in Python [26], and Prophet is available as a library [11]. The difficulty for a small business is therefore not access to algorithms, but choosing, running, evaluating and explaining them automatically, which is what the component does.

2.5 Gap synthesis and conceptual framework

Table 2.4 brings the review together into the four gaps stated in Section 1.4 and shows how this thesis addresses each.

Table 2.4: Research gaps and how this thesis addresses them

Gap	Evidence from the review	How this thesis responds	RQ
G1 Short and growing histories	Competitions and studies use long or daily series [3]–[5]; minimum-history rules are unclear [19]	A model ladder selected by months of history, evaluated separately for each history band	RQ1
G2 Uncertainty for non-experts	Intervals rarely evaluated in restaurant studies [6]; numerical ranges do not damage trust much [16]	95% ranges evaluated for coverage and width, and shown with accuracy and a data-quality message	RQ2, RQ3
G3 Integration into everyday tools	Studies report offline accuracy [1], [13]	A forecast embedded in a multi-tenant POS with authentication and branch isolation	O5
G4 Local context	POS products in Sri Lanka offer historical reports (Table 2.3)	Monthly sales, cost and profit forecasts in LKR inside a Sri Lankan POS	RQ3

The conceptual framework of the study links these elements. The **inputs** are the orders and expenses that a branch already records in its POS. The **data stage** turns them into complete monthly series and measures how much history exists. The **model stage** chooses the model that the history can support, from a moving average to damped Holt-Winters, following the evidence that simple, damped methods are robust on short series [10], [12]. The **output stage** gives the owner a forecast, a range, an accuracy figure and an explanation of the data. The **evaluation** asks whether each stage works: rolling-origin accuracy against baselines (RQ1), coverage

of the ranges (RQ2) and the owners' understanding and perceived usefulness (RQ3). Figure 4.3 in Chapter 4 shows the model stage in detail.

3 RESEARCH METHODOLOGY

This chapter explains how the research was carried out: the overall design, the datasets, the experimental procedure, the metrics, the analysis, the ethical considerations and how the results can be reproduced.

3.1 Research design

The study follows design science research, in which an artefact is built to address a practical problem and is then evaluated rigorously [27]. The artefact is the forecasting component described in Chapters 4 and 5. The evaluation is mixed-method: a quantitative experiment measures forecast accuracy and the reliability of the prediction ranges, and a qualitative user study examines whether owners can understand and use the output. Table 3.1 maps each research question to its method.

Table 3.1: Research questions, methods and evidence

RQ	Method	Data	Evidence
RQ1 Accuracy	Rolling-origin backtest against baselines	S1–S3 (Table 3.2)	MAPE, MAE, RMSE, MASE by horizon and history length; paired tests
RQ2 Ranges	Coverage analysis of the same backtest	S1–S3	Share of actuals inside the 95% range; range width
RQ3 Usefulness	Task-based user study with questionnaire and interview	Restaurant owners and managers	SUS score, Likert items, interview themes
Robustness	Case analysis	S4	Forecast and backtest error with and without unrecorded months

3.2 Data, participants or study materials

Four datasets were used (Table 3.2). SmartCost has been in live use at three businesses only since September 2026 (Section 5.5), too short a history to evaluate, so the main experiment uses simulated branches whose generating process is known. This has a methodological advantage: because the seasonal pattern and growth rate are set by the generator, it is possible to check whether a model recovers them.

Table 3.2: Datasets used in the evaluation

ID	Dataset	Series	Length	Purpose
----	---------	--------	--------	---------

ID	Dataset	Series	Length	Purpose
S1	Simulated branches from a port of the SmartCost seed generator (order-level simulation, Table 5.2)	30 branches $\times$ sales and costs	36 complete months	Main experiment
S2	S1 with an additional month-level demand shock, uniform $\pm 12\%$	30×2	36 months	Sensitivity to irregular months
S3	Development branch history generated by the real JavaScript seed and read from MongoDB	1×2	29 months (Mar 2024–Jul 2026)	Check against system-produced data
S4	The same branch as the app sees it, including a half-recorded month and a month with 3 test orders	1×2	31 months	Robustness case

The Python port reproduces the generator's rules exactly (Table 5.2): daily order counts from a base of 18 orders scaled by growth, month-of-year and weekday multipliers and daily noise; between one and four items per order drawn from the 20-item menu by popularity weight; and the same fixed, variable and irregular costs. Each branch uses its own random seed (1000 to 1029) and a start month that rotates through the calendar, so that the evaluation is not tied to one seasonal phase.

For the user study, 5 to 10 owners or managers of small restaurants and cafés in Sri Lanka are to be recruited. Each participant's role, type of business and years of operation will be recorded without identifying details.

3.3 Experimental or study procedure

The forecasting experiment uses rolling-origin evaluation, which tests a method on many forecast origins rather than one holdout period [14]. For each series of N complete months:

1. For every origin t from 3 to $N - 1$, the first t months are used for training.
2. The production function `forecast_series` from `forecasting.py` produces forecasts and 95% ranges for the next 12 months, exactly as it does for the app.
3. Each baseline produces forecasts from the same training data (Table 3.3).
4. Every forecast for which the actual value is known is recorded with its origin, horizon, actual and predicted values and range.
5. Errors are summarised by horizon, by length of history and by method, as described in Section 3.5.

Table 3.3: Methods compared

Method	Definition	Role
Production ladder	The component as deployed: moving average (3 months), damped Holt (4–23), damped Holt-Winters (24+)	Artefact under test
Naive	Every future month equals the last observed month	Baseline
Seasonal naive	Every future month equals the same month one year earlier (needs 12 months)	Standard seasonal benchmark [2]
3-month moving average	Mean of the last three months	Baseline
Linear trend	Least-squares straight line through the history	Baseline
Damped Holt only	The ladder without its seasonal rung	Ablation: value of seasonality
Revised ladder	Moving average below 12 months, seasonal naive for 12–23, production model from 24	Exploratory variant derived from the results

The revised ladder was added after the first analysis showed where the production ladder lost to simpler methods. It is reported as an exploratory result, because its thresholds were chosen from the same data on which it is evaluated.

The user study follows a fixed protocol. After a short explanation of the Forecast section, each participant completes three tasks on a phone with seeded data: (1) read the expected sales for the next three months and their range, (2) decide whether costs are expected to rise or fall, and (3) explain how reliable the forecast is. The participant then completes the System Usability Scale [18] and five usefulness items based on the technology acceptance model [17], followed by a ten-minute semi-structured interview. The instruments are in Appendix A.

3.4 Measures and evaluation metrics

Accuracy is measured with the metrics defined in Equations 4.6 to 4.8 and with the mean absolute scaled error (MASE), which divides each absolute error by the in-sample mean absolute error of the seasonal naive method (or of the naive method when the history is shorter than 13 months). MASE is scale-free and is defined when actual values are zero, which addresses known weaknesses of MAPE [15]. A MASE

below 1 means the method beats the in-sample benchmark. Table 3.4 lists all measures.

Table 3.4: Measures and evaluation metrics

Measure	What it captures	Used for
MAPE (%)	Average error relative to the actual value	RQ1; the measure owners understand
MAE, RMSE (LKR)	Absolute error; RMSE penalises large misses	RQ1
MASE	Error relative to a naive benchmark on the training data	RQ1
Error of horizon totals (%)	Error of the 3- and 12-month totals shown on the app's cards	RQ1
Coverage (%)	Share of actual values inside the 95% range	RQ2
Range width (% of actual)	Width of the range relative to the actual value	RQ2
Response time (ms)	Time to compute a forecast	NFR2
SUS score (0–100), Likert items, themes	Usability and perceived usefulness	RQ3

3.5 Data analysis

Errors are aggregated in three ways. **By history length**, folds are grouped by the number of training months (3–5, 6–11, 12–17, 18–23 and 24–35) using forecasts one to three months ahead, which directly tests the design of the ladder. **By horizon on common folds**, all methods are compared only on origins with at least 12 months, where every baseline, including the seasonal naive, can produce a forecast; comparing methods on different folds would favour those that skip the hardest short-history cases. **By horizon totals**, the error of the summed 3- and 12-month forecasts is measured, because these are the figures the app shows on its cards.

Pairs of methods are compared with the Wilcoxon signed-rank test on the absolute errors of the same folds, which does not assume normally distributed errors. A difference is treated as significant at the 5% level. Because neighbouring origins of the same branch overlap, the folds are not fully independent, so p-values are read as

supporting evidence alongside the share of folds in which one method is more accurate. Interview notes are analysed with thematic coding.

3.6 Ethics, privacy, security and responsible AI

- **No personal data in the experiment.** The forecasting evaluation uses simulated data and the development system's seeded data only.
- **User study.** Participation is voluntary and based on informed consent (Appendix A); responses are anonymised as P1, P2 and so on, and no business financials are collected. Whether SLIIT ethics approval is required is confirmed with the supervisor before recruitment.
- **Privacy and security in the product.** A forecast is computed only for the shop and branch in the user's authenticated token, the analysis service is not exposed to the internet, and it receives only anonymous monthly totals, never customer records.
- **Responsible use of prediction.** The app always shows a range and a plain-language data-quality message next to the forecast, and reports the measured accuracy, so that the owner is not led to over-trust a single number. The model is a transparent statistical method whose components can be explained.
- **Use of AI tools.** Generative AI assistance used in this project is disclosed in Appendix B.

3.7 Reproducibility and research artefacts

All artefacts are in the project repository. The evaluation is deterministic: each simulated branch uses a fixed NumPy random seed, and re-running the scripts reproduces every number in Chapter 6. Table 3.5 lists the commands.

Table 3.5: Commands to reproduce the results (run in analysis-backend)

Step	Command	Output
Unit tests	<code>python -m pytest tests</code>	14 passing tests
Evaluation	<code>python -m experiments.evaluate_forecasting</code>	<code>results/fold_forecasts.csv</code> , <code>results/summary.json</code>
Worked examples	<code>python -m experiments.example_forecasts</code>	<code>results/examples.json</code>

Step	Command	Output
Figures (needs matplotlib)	<code>python -m experiments.make_figures</code>	<code>results/figures/*.png</code>

The software versions are pinned in `requirements.txt` (statsmodels 0.14.6, NumPy 2.5.2, SciPy 1.18.0, FastAPI 0.141.1) and `requirements-dev.txt` (pytest 9.1.1). The backend tests run with `npm test` in backend.

The evaluated version of the forecasting code, tests and evaluation scripts is commit `fe210364` (`fe210364a10cf1b2ad6480936e52dcbbcf5ece22`) on the main branch of the project repository.

4 PRODUCT AND SOLUTION DESIGN

This chapter turns the problem and objectives of Chapter 1 into a concrete design for the sales, cost and profit forecasting component of SmartCost POS. It states the requirements the component must meet, places the component within the complete SmartCost architecture, explains the main design decisions together with the alternatives that were rejected, and then describes the detailed design of the data pipeline, the forecasting algorithm, the application programming interface (API) and the user interface.

4.1 Requirements

The requirements were derived from three sources: the problem statement in Section 1.3, the planning needs of small restaurant owners identified during requirement gathering, and the constraints of the existing SmartCost platform, in which every sale is stored in the History collection and every business expense in the CostExpense collection. Table 4.1 lists the functional requirements and Table 4.2 the non-functional requirements. Each requirement is traced to the test that verifies it in Section 5.4.

Table 4.1: Functional requirements of the forecasting component

ID	Requirement	Priorit y	Verified by
FR1	Aggregate completed sales and recorded expenses into calendar-month totals for one shop branch, using Sri Lankan time (Asia/Colombo).	High	TC-N1, TC-A3
FR2	Exclude the month in progress from model training, but report its actual-to-date values and project its month-end total.	High	TC-A3
FR3	Select the forecasting model automatically from the number of complete months of history.	High	TC-P1 to TC-P4
FR4	Forecast sales and costs for each of the next 12 months and derive profit and profit margin.	High	TC-A3
FR5	Summarise the forecast for 1, 3, 6 and 12-month horizons, each with a 95% prediction range.	High	TC-N2, TC-P6
FR6	Report the accuracy of each model (MAPE, RMSE, MAE) and a holdout backtest.	Medium	TC-P8, TC-P9

ID	Requirement	Priority	Verified by
FR7	Explain in plain language how much history the forecast is based on and how reliable it is.	Medium	TC-N3, TC-S2
FR8	Present the forecast in the mobile POS app with a horizon selector, chart and monthly breakdown.	High	TC-S1 to TC-S4

Table 4.2: Non-functional requirements of the forecasting component

ID	Category	Requirement
NFR1	Security	Only an authenticated user may request a forecast, and only for the shop and branch in that user's session token.
NFR2	Performance	A forecast must be returned well within the 25-second timeout of the analysis-service client; the target is under 2 seconds for 36 months of history.
NFR3	Determinism	The same history must always produce the same forecast, so that owners and examiners can reproduce results.
NFR4	Usability	A non-technical owner must be able to read the forecast, its range and its reliability without statistical training.
NFR5	Maintainability	The component must be isolated in its own folders so that it can be developed in parallel with the other three research components without merge conflicts.
NFR6	Robustness	Short, sparse or irregular histories must degrade to a simpler model or a clear message, never to an error.

Two assumptions underlie these requirements. First, a branch records its sales through the SmartCost checkout, so History is a complete record of revenue; second, expenses are entered in the cost-management module with a purchase date. Where either assumption fails, the forecast inherits the gap, which is why FR7 requires the component to report the amount of history it used.

4.2 Component architecture within the complete solution

SmartCost POS consists of an Express/MongoDB backend, a Next.js web dashboard, an Expo/React Native mobile POS application and, since the research components were added, a Python FastAPI analysis service. Figure 4.1 shows where the forecasting component sits in this architecture and which parts it shares with the core platform and with the other three research components.

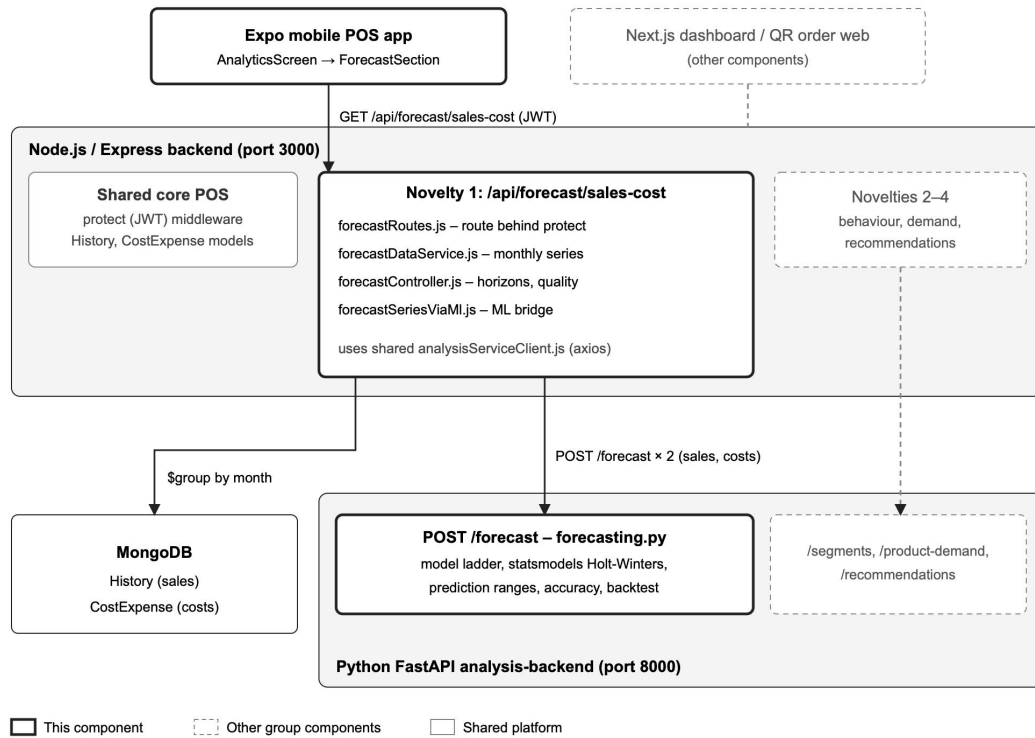


Figure 4.1: Architecture of the forecasting component within SmartCost POS

The component follows a three-layer design. The **presentation layer** is the Forecast section of the Analytics screen in the mobile app. The **orchestration layer** runs in Node.js: it authenticates the request, reads the branch's transactions from MongoDB, turns them into monthly series and assembles the final response. The **modelling layer** runs in the Python analysis service, which fits the forecasting models and returns predictions, ranges and accuracy figures. Node.js never performs model arithmetic, and Python never accesses the database. This separation keeps business data, authentication and tenancy rules in one place, while placing the statistics in the language with the most mature time-series libraries.

The component reuses three parts of the core platform: the protect authentication middleware, the History and CostExpense data models, and the Analytics screen of the mobile app. It introduced the Python analysis service and its Node.js client, `analysisServiceClient.js`, which the customer behaviour, product demand and recommendation components later adopted for their own models (Section 5.3).

4.3 Design alternatives and decisions

Table 4.3 summarises the main design decisions. For each, the options considered, the option chosen and the reason are given. The decisions with the largest effect on the research outcome are then discussed in more detail.

Table 4.3: Design decisions and alternatives considered

Decision	Options considered	Chosen	Reason
Where models run	Hand-written JavaScript in Node.js; Python service	Python FastAPI service with statsmodels	Optimised parameter fitting and a well-tested library instead of a hand-rolled grid search; reusable by other components.
Time granularity	Daily, weekly, monthly	Monthly	Matches how owners plan stock, staff and cash; smooths daily noise; keeps series short enough to fit quickly.
Model family	ARIMA/SARIMA, Prophet, LSTM, exponential smoothing	Exponential smoothing ladder	Works on short series, is interpretable, and performs strongly in forecasting competitions [3].
Short histories	Refuse to forecast; one model for all; ladder by data size	Ladder by data size	A new branch receives a usable forecast from its third complete month.
Trend form	Linear trend; damped trend	Damped additive trend	Prevents 12-month forecasts from extrapolating a recent slope indefinitely [9].
Seasonality form	Additive; multiplicative	Additive, period 12	Numerically stable on short series and allows months with very low values.
Profit	Model profit directly; derive it	Forecast sales and costs separately; profit = sales – costs	Sales and costs follow different patterns (seasonal demand against largely fixed rent and salaries).
Prediction range	Model-based simulation; residual heuristic	$1.96 \times \text{residual SD} \times \sqrt{h}$	Simple, model-agnostic and explainable; its coverage is evaluated empirically in Chapter 6.
Horizon totals	Combine variances; sum the monthly bounds	Sum the monthly bounds	Deliberately conservative: assumes errors in consecutive months may not cancel out.

Decision	Options considered	Chosen	Reason
Python unavailable	JavaScript fallback model; fail	Return HTTP 500	A single source of model truth; the trade-off is discussed in Section 5.5.

4.3.1 Exponential smoothing rather than ARIMA or machine learning

A typical SmartCost branch has between a few months and three years of history, that is, at most 36 monthly observations. ARIMA-family models [2] need careful differencing and order selection that is unreliable on so few points, and deep-learning models such as LSTM networks require far more data than a single small restaurant produces; on 1,045 monthly competition series, machine-learning methods were less accurate than simple statistical methods at every horizon [12]. Exponential smoothing models, by contrast, are defined by a small number of smoothing parameters, can be fitted on short series, and decompose a forecast into level, trend and seasonality, which can be explained to an owner [2]. Large forecasting competitions have also shown that such simple statistical methods remain difficult to beat on business series [3].

4.3.2 A model ladder chosen by the amount of history

No single model suits every branch. A seasonal model cannot estimate twelve monthly effects from ten months of data, while a moving average wastes the information in a long history. The component therefore selects one of four behaviours from n , the number of complete months available:

- $n < 3$: no forecast is produced; the user receives a message explaining how much more history is needed.
- $n = 3$: a three-month moving average, which assumes only a stable level.
- $4 \leq n < 24$: Holt's linear method with a damped trend, which adds a trend but no seasonality.
- $n \geq 24$: additive Holt-Winters with a damped trend and a 12-month season.

The threshold of 24 months reflects the common requirement of at least two complete seasonal cycles before seasonal components can be separated from noise. The threshold of 3 months is the smallest history from which a level can be estimated

with any meaning. These thresholds are themselves a research variable: Chapter 6 evaluates the accuracy of each rung of the ladder at different history lengths.

4.3.3 Excluding the month in progress

A month that is only partly complete would appear to the model as a sudden collapse in sales and costs. The data pipeline therefore flags the current month as partial and trains only on complete months. The partial month is still useful to the owner, so the model is asked for one extra step ahead and that first step is reported as the projected month-end total next to the actual figures recorded so far.

4.4 Detailed design

4.4.1 Data sources and monthly aggregation

Table 4.4 shows the fields read by the component. Sales are taken only from orders whose status is submitted (the spelling used throughout the SmartCost data model), so that open carts and cancelled orders are excluded. Both collections already carry compound indexes on shop, branch and date, so the monthly aggregation does not require new indexes.

Table 4.4: Data fields used by the forecasting component

Collection	Field	Use
History	shopId, branchId	Tenant and branch filter, taken from the session token
History	status	Only submitted (completed) orders are counted
History	checkOutTime	Assigns the sale to a calendar month in Asia/Colombo time
History	totalAmount	Summed to give monthly sales
CostExpense	shopId, branchId	Tenant and branch filter
CostExpense	purchaseDate	Assigns the expense to a calendar month
CostExpense	amount	Summed to give monthly costs

The aggregation looks back 36 months, groups each collection by year and month, and joins the two results into one row per calendar month. Months with no activity are filled with zero so that the series is evenly spaced, and empty months before the

branch's first sale or expense are removed so that a new branch is not penalised for months in which it did not yet exist.

4.4.2 Request flow

Figure 4.2 shows the sequence of interactions for one forecast request. The two calls to the analysis service, one for the sales series and one for the cost series, are issued in parallel. Because model fitting is CPU-bound, a single analysis-service worker still computes the two fits one after the other, so the parallel calls shorten the response only when the service runs more than one worker (Section 6.2).

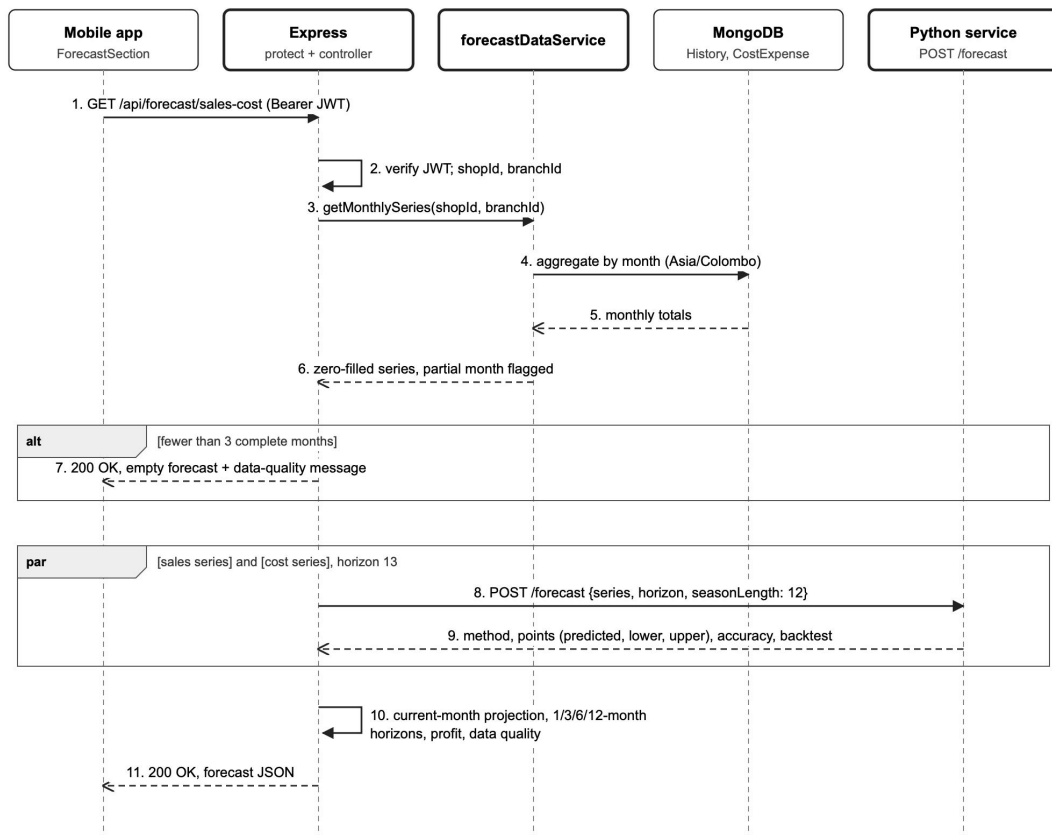


Figure 4.2: Sequence of interactions for one forecast request

4.4.3 Model selection and forecasting algorithm

Figure 4.3 shows the model-selection logic of the analysis service, including the fallback chain that is followed if a model fails to fit.

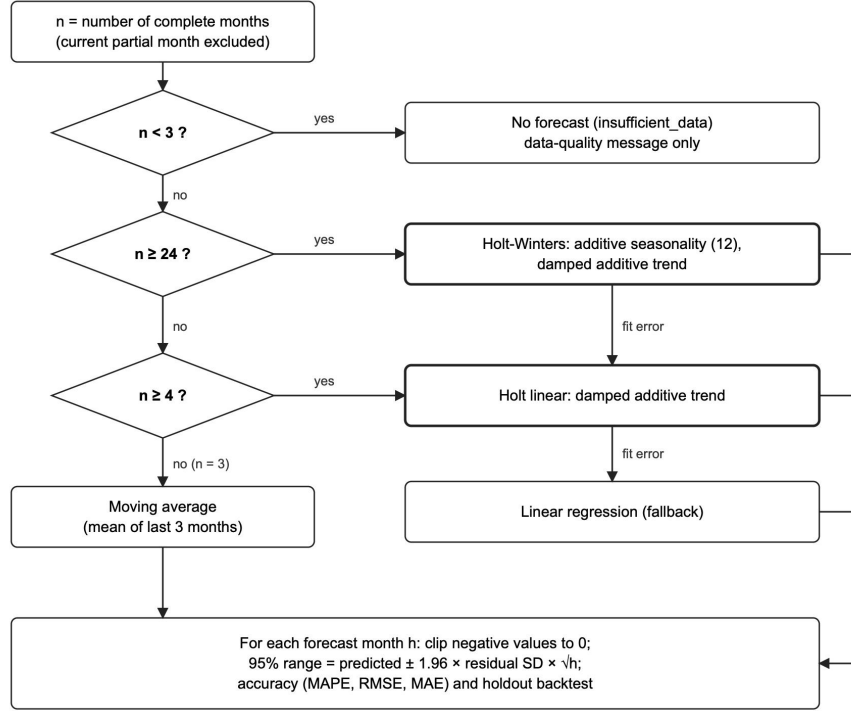


Figure 4.3: Model selection ladder and fallback chain

For a series y with $n \geq 24$ complete months, the component fits an additive Holt-Winters model with a damped trend. Using the notation of Hyndman and Athanasopoulos [2], with level ℓ , trend b , seasonal component s , season length $m = 12$ and smoothing parameters α , β , γ and damping parameter ϕ , the model is:

$$\ell_t = \alpha (y_t - s_{t-m}) + (1 - \alpha) (\ell_{t-1} + \phi b_{t-1}) \quad (4.1)$$

$$b_t = \beta (\ell_t - \ell_{t-1}) + (1 - \beta) \phi b_{t-1} \quad (4.2)$$

$$s_t = \gamma (y_t - \ell_{t-1} - \phi b_{t-1}) + (1 - \gamma) s_{t-m} \quad (4.3)$$

$$\hat{y}_{t+h|t} = \ell_t + (\phi + \phi^2 + \dots + \phi^h) b_t + s_{t+h-m(k+1)} \quad (4.4)$$

where k is the integer part of $(h - 1)/m$. Holt's damped linear method is the same model without the seasonal equation. The parameters are estimated by the statsmodels optimiser (L-BFGS-B), which minimises the in-sample sum of squared one-step-ahead errors; under Gaussian errors this is equivalent to maximising the likelihood [26]. Forecasts below zero are clipped to zero, because negative sales or costs are not meaningful.

Each forecast month h is given a 95% prediction range based on the standard deviation $\hat{\sigma}$ of the in-sample residuals, widened with the square root of the horizon (the $\sqrt{h}$ factor is exact for the naive method and an approximation for the damped and seasonal models [2]):

$$\hat{y}_{t+h} \pm 1.96 \hat{\sigma} \sqrt{h} \quad (4.5)$$

This heuristic treats forecast errors as accumulating like a random walk. It is not the exact interval of any one model, which is why its empirical coverage is measured in Chapter 6 rather than assumed.

Accuracy is reported with three metrics (Equations 4.6 to 4.8), where $e_t = y_t - \hat{y}_t$ is the error in month t and N is the number of months compared. The mean absolute percentage error skips months with zero actual value, for which it is undefined:

$$\text{MAE} = \frac{1}{N} \sum |e_t| \quad (4.6)$$

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum e_t^2} \quad (4.7)$$

$$\text{MAPE} = \frac{100}{N} \sum \left| \frac{e_t}{y_t} \right| \quad (4.8)$$

The service computes these metrics twice: on the fitted values over the training data (in-sample accuracy), and on a holdout backtest. The backtest holds out the last $\min(6, \lfloor n/4 \rfloor)$ months, refits the ladder on the remaining months and compares its forecast with the held-out actual values. When the seasonal model is used, the holdout is further limited so that at least 24 months remain for training. A consequence of this rule is that a series of exactly 24 months has no backtest, because no month can be held out without losing the seasonal model.

4.4.4 Interface design

The component exposes one endpoint to the mobile app and uses one endpoint of the analysis service. Table 4.5 summarises both.

Table 4.5: API endpoints of the forecasting component

Endpoint	Caller	Input	Output
----------	--------	-------	--------

Endpoint	Caller	Input	Output
GET /api/forecast/sales-cost	Mobile app (JWT required)	None; shop and branch are taken from the token	history, currentMonth, forecast.months, forecast.horizons, models (method, parameters, accuracy, backtest) and dataQuality
POST /forecast	Node.js backend	series (list of numbers), horizon (integer), seasonLength (default 12)	method, params, points (predicted, lower, upper), accuracy, backtest

The dataQuality object carries one of four levels, chosen from the number of complete months, together with a plain-language message that the app displays above the forecast (Table 4.6).

Table 4.6: Data-quality levels reported to the user

Level	Complete months	Message shown to the owner (abridged)
none	0	No completed months of sales history yet; record sales for at least 3 months.
insufficient	1–2	Only n complete month(s) of history; at least 3 are needed.
limited	3–23	Based on n months; seasonal patterns need 24 months, so longer horizons are less reliable.
good	24 or more	Based on n complete months of history, including seasonal patterns.

4.4.5 User interface design

The Forecast section is designed for an owner who is not trained in statistics. Its main elements, from top to bottom, are:

- a **horizon selector** with the options Next month, Next 3 months (the default), Next 6 months and Next 1 year;
- **prediction cards** showing forecast sales, costs and profit for the selected horizon, each with its range and the profit margin;
- a **line chart** of the last six complete months of actual sales and costs followed by six forecast months, with the forecast drawn in a distinct colour;

- a **monthly breakdown** listing each forecast month in the selected horizon;
- a **current-month card** comparing actual sales and costs so far with the projected month-end totals;
- a **model accuracy panel** showing the backtest MAPE for sales and costs and the name of the model in plain words, for example "Holt-Winters (seasonal, damped trend)"; and
- the **data-quality message** from Table 4.6.

Amounts are formatted in Sri Lankan rupees (for example, Rs. 1,250,000), and large values on the chart axis are abbreviated (1.3M, 450K). Figure 4.4 shows the top of the section, with its header and the horizon selector.

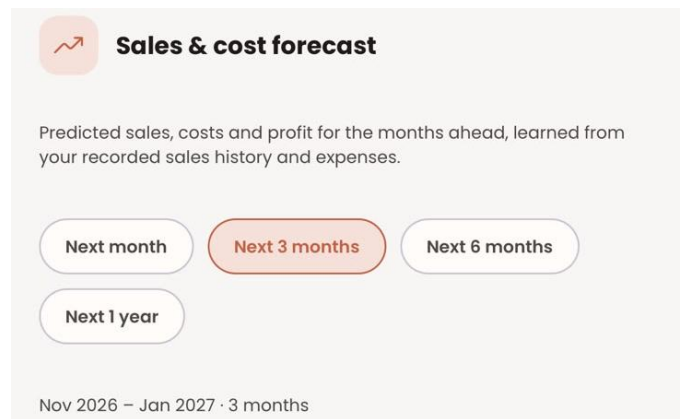


Figure 4.4: Header and horizon selector of the forecast section

Figures 4.5 and 4.6 show the implemented section on an iPhone for the development branch after 36 complete months of data had been seeded (4 October 2026). For the next three months it forecasts sales of Rs. 8.88 million (range 7.8–9.9 million), costs of Rs. 4.12 million and a profit of Rs. 4.76 million. The December peak learned from earlier years is visible both in the chart and in the monthly breakdown, where December 2026 (Rs. 3.51 million) is well above November (Rs. 2.81 million) and January (Rs. 2.55 million).

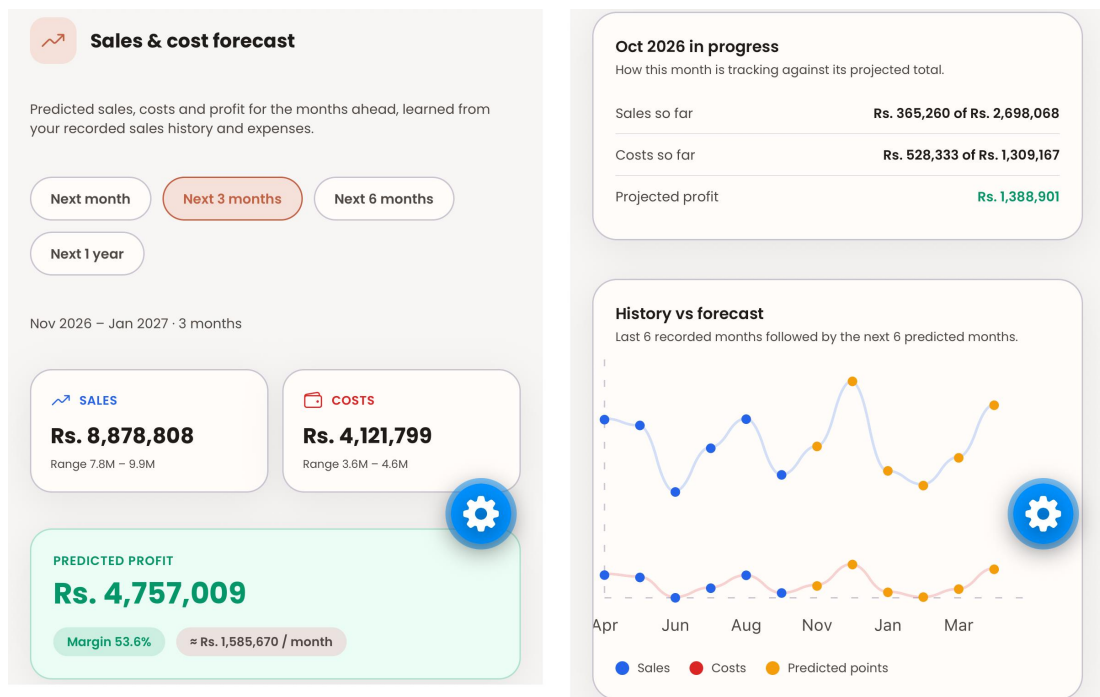


Figure 4.5: Forecast section for the development branch with 36 months of seeded data: (a) prediction cards for the next three months; (b) month in progress and History vs forecast chart

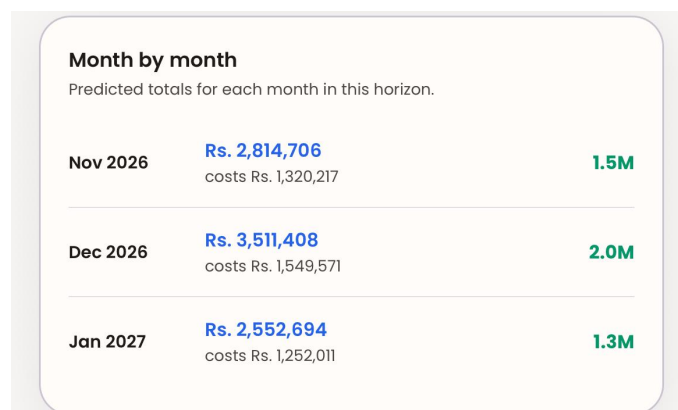


Figure 4.6: Month by month breakdown of the three-month forecast

Figure 4.7 shows the model accuracy panel for the same branch. The errors are backtest MAPEs: the model is refitted without the last six complete months and scored on them, giving 4.99% for sales and 4.7% for costs with the Holt-Winters model and 35 months of history.

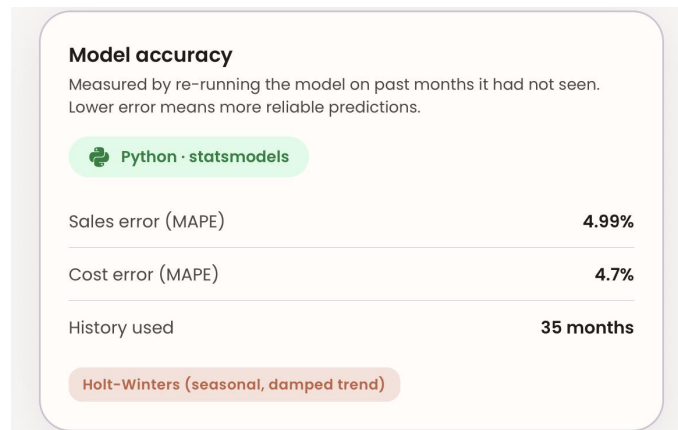


Figure 4.7: Model accuracy panel with backtest errors, history length and model name

4.5 Standards and professional practice

The design follows recognised standards and practices wherever they apply:

- **REST conventions:** resources are read with GET and computed with POST, and HTTP status codes signal the outcome (200, 400, 401, 500).
- **JSON Web Tokens (RFC 7519)** carry the user's shop and branch, so the tenant is never taken from the request body.
- **Schema validation:** every request to the analysis service is validated by Pydantic models before any computation.
- **ISO 8601** dates in API responses and **ISO 4217** currency (LKR) in the user interface.
- **Requirements engineering:** requirements are uniquely identified and traceable to tests, following the practices of ISO/IEC/IEEE 29148.
- **Secure API design** informed by the OWASP API Security Top 10, in particular object-level authorisation (a user cannot request another branch's forecast).
- **Version control:** each group member developed on a personal branch (IT22544872_chamath for this component) that was merged into main.
- **Coding style:** PEP 8 for Python and the project's ESLint configuration for JavaScript and TypeScript.

5 IMPLEMENTATION, INTEGRATION AND TESTING

This chapter describes how the design of Chapter 4 was implemented, how the component was integrated with the rest of SmartCost POS and with the other research components, how it is verified, and how it is deployed and operated.

5.1 Development environment and technology stack

Table 5.1 lists the technologies used. Library versions are pinned in analysis-backend/requirements.txt and in the package.json files of the backend and mobile app, so the environment can be reproduced exactly.

Table 5.1: Technology stack of the forecasting component

Layer	Technology	Purpose
Mobile app	Expo, React Native, TypeScript, React Native Paper, react-native-chart-kit	Forecast screen and chart
Backend	Node.js 20+, Express, Mongoose, axios	Authentication, aggregation, orchestration
Database	MongoDB	History and CostExpense collections
Analysis service	Python 3.14, FastAPI 0.141.1, Pydantic 2.13.4, uvicorn	Forecast API and validation
Modelling	statsmodels 0.14.6, NumPy 2.5.2	Exponential smoothing and metrics
Tools	Visual Studio Code, Git and GitHub, Postman, pytest 9.1, Jest 30.5, supertest 7.3	Development, version control, API testing

5.2 Component implementation

The component consists of five modules, described below in the order in which data flows through them.

5.2.1 Monthly data preparation

forecastDataService.js builds the monthly series. It runs two MongoDB aggregation pipelines in parallel, one over History and one over CostExpense. Figure 5.1 shows the sales pipeline: completed orders for the branch are matched, then grouped by year

and month in the Asia/Colombo time zone, so that a sale at 11:30 pm on the last day of a month is counted in that month and not in the next one, as it would be in UTC.

```
const rows = await History.aggregate([
  { $match: { shopId, branchId, status: 'submitted',
    checkOutTime: { $gte: rangeStart } } },
  { $group: {
    _id: {
      year: { $year: { date: '$checkOutTime', timezone: 'Asia/Colombo' } },
      month: { $month: { date: '$checkOutTime', timezone: 'Asia/Colombo' } },
    },
    totalAmount: { $sum: '$totalAmount' },
    recordCount: { $sum: 1 },
  } },
]);
```

Figure 5.1: Monthly sales aggregation in forecastDataService.js (simplified)

The service then walks month by month through the 36-month window, fills months without activity with zero, marks the current month as partial, and drops leading empty months. Each row carries sales, costs, profit and the number of orders and expenses behind them.

5.2.2 Orchestration

forecastController.js handles the request. It first checks that the token contains both a shop and a branch, and returns HTTP 400 with the code `BRANCH_REQUIRED` if no branch has been selected. If fewer than three complete months exist it returns an empty forecast with the appropriate data-quality message, without calling the analysis service. Otherwise it sends the sales and cost series to the analysis service in parallel with a horizon of 13 months: step 0 becomes the projection for the current partial month, and steps 1 to 12 become the forecast months. The controller then labels each month, derives profit, builds the four horizon summaries and attaches the data-quality assessment.

5.2.3 Bridge to the analysis service

forecastSeriesViaML.js forwards a series to the analysis service through the shared analysisServiceClient.js, an axios instance whose base URL and timeout (25 seconds by default) come from the `ANALYSIS_SERVICE_URL` and `ANALYSIS_SERVICE_TIMEOUT_MS` environment variables. It tags each result with `poweredBy: 'python'` so that the response records which engine produced it.

5.2.4 Forecasting engine

forecasting.py implements the ladder of Figure 4.3. Figure 5.2 shows the selection function and the construction of the prediction ranges. Each model function catches fitting errors and falls back to the next simpler model, so that an unusual series never produces an error for the user.

```
def _select_model(series, horizon, season_length):
    n = len(series)
    if n >= max(24, 2 * season_length):
        return _holt_winters_forecast(series, horizon, season_length)
    if n >= 4:
        return _holt_linear_forecast(series, horizon)
    return _moving_average_forecast(series, horizon)

residual_spread = _std(series - fitted)
for h, raw in enumerate(model["forecast"], start=1):
    predicted = max(raw, 0)
    margin = 1.96 * residual_spread * math.sqrt(h)
    points.append({"predicted": predicted,
                  "lower": max(predicted - margin, 0),
                  "upper": predicted + margin})
```

Figure 5.2: Model selection and prediction ranges in forecasting.py (simplified)

The Holt-Winters and Holt models are fitted with `ExponentialSmoothing(...).fit(optimized=True)` from `statsmodels`, with `damped_trend=True`. The FastAPI router validates the request against the `ForecastRequest` Pydantic model and returns HTTP 500 with the error text if computation fails.

5.2.5 Mobile user interface

`ForecastSection.tsx` implements the screen designed in Section 4.4.5 and is rendered inside the existing `AnalyticsScreen`. Its data access is kept in a separate `forecastService.ts` that calls the endpoint directly rather than through the shared Redux store. This choice was deliberate: it meant the component added no changes to shared state code, which kept merges with the other members' branches free of conflicts. The service checks for an internet connection first and converts API failures into a readable message with a retry option.

5.2.6 Synthetic data generator

Because real multi-year restaurant data was not available at the start of the project, a reproducible data generator, `seedForecastData.js`, was built for development and evaluation. It uses a fixed pseudo-random seed, so every run with the same options produces an identical dataset. Table 5.2 lists its parameters. The seasonal pattern is

known in advance, so the generator also serves as ground truth for checking whether the model recovers that pattern.

Table 5.2: Parameters of the synthetic restaurant data generator

Parameter	Value
Random seed (default)	20260816
History length	24 months by default, including the current partial month; set with --months
Base demand	18 orders per day, 1–4 items per order, 1–3 units per item, 20 Sri Lankan menu items
Growth	0.6% per month
Monthly seasonality (Jan–Dec)	0.88, 0.90, 0.97, 1.18, 1.00, 0.92, 1.02, 1.10, 0.95, 0.98, 1.05, 1.28
Weekday seasonality (Sun–Sat)	1.20, 0.85, 0.85, 0.90, 1.00, 1.25, 1.40
Daily noise	Uniform multiplier between 0.82 and 1.18
Fixed costs	Rent LKR 85,000 per month; salaries LKR 220,000 and utilities LKR 38,000, both growing with demand and varying slightly
Variable costs	Ingredients at 32% of monthly sales, bought in four weekly purchases
Irregular costs	Marketing LKR 12,000–45,000 in 55% of months; maintenance LKR 8,000–60,000 in 40% of months

Note that the default of 24 months includes the month in progress, so it yields only 23 complete months and exercises the damped Holt rung of the ladder. The seasonal rung requires --months=25 or more; the evaluation in Chapter 6 uses --months=36, the maximum the aggregation reads.

5.3 Integration with the other components

The forecasting component was the first of the four research components to be completed, and its structure became the pattern for the others. Integration took place at four points:

1. **Shared analysis service.** The FastAPI application in `analysis-backend/app/main.py` mounts one router per research component. The forecasting router provides POST

/forecast; the customer behaviour, product demand and recommendation components add /segments, /product-demand and /recommendations to the same service, which also exposes a /health endpoint.

2. **Shared client.** analysisServiceClient.js holds one axios instance and one function per analysis endpoint, so the timeout and base URL are configured once for all four components.
3. **Shared platform.** The component relies on the core protect middleware, the History and CostExpense models and the mobile Analytics screen, and changes none of them.
4. **Folder and naming convention.** Each component lives in its own novelty0X folder in the backend, the analysis service and the mobile app, with the same controller, data-service, routes and ML-bridge split. This allowed the four members to work on separate Git branches and merge into main with few conflicts.

The component's routes are registered in backend/src/app.js under /api/forecast, alongside the routes of the core POS and the other components.

Figure 5.3 shows the Analytics screen of the mobile app with real branch data. The forecast section is appended below the existing period summary, so the owner sees what has happened and what is expected on one screen, without a separate navigation entry.

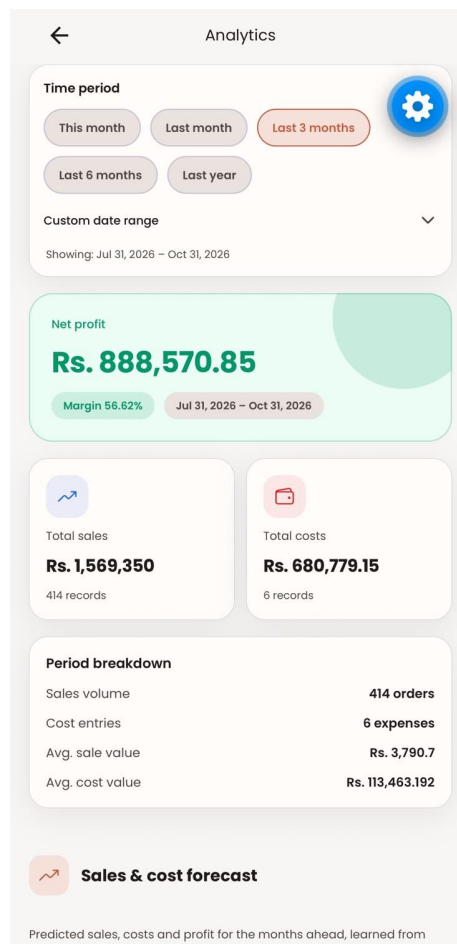


Figure 5.3: Analytics screen of the mobile app, with the forecast section below the period summary

The member branches IT22151810_kasun and IT22162250_Dasitha were merged into main on 31 August 2026. The merge caused two build defects, both outside the forecasting component. First, `fontend/package.json` still contained unresolved conflict markers from a stashed change, so the app dependencies could not be installed; they were removed and the `@lottiefles/dotlottie-react` dependency kept (commit 14b55202). Second, merged code imported a helper, `safeSecureStorage.ts`, that had not been committed, so the app did not build; the helper was added together with the updated lock file (commit dfa5d8e7). One merged change touched this component directly: to reuse the monthly series in another chart, `getMonthlySeries` gained an optional `keepEmptyMonths` flag. Its default leaves the forecast's behaviour unchanged, which the tests in Section 5.4 confirm on the merged code.

5.4 Verification and software testing

Verification was planned at four levels: unit tests of the forecasting engine, unit tests of the Node.js data and controller logic, API integration tests, and system tests of the mobile interface. Table 5.3 lists the test cases and the requirement each one verifies.

The automated tests are in two suites. The Python suite, `analysis-backend/tests/test_forecasting.py`, runs with `pytest` against the production `forecasting.py` and, through FastAPI's test client, against the `POST /forecast` endpoint. Its test series are generated with the same month-of-year multipliers as the synthetic data generator, from a fixed random seed. The Node.js suite, `backend/tests/forecast.test.js`, runs with `Jest` and `supertest`. It mounts the real route, protect middleware and controller on an Express application, and replaces only the User model, the MongoDB aggregation and the analysis-service client with in-memory fakes, so the tests need neither a database nor the Python service. A third file, `backend/tests/forecast.live.test.js`, checks the contract between the two services against a running analysis service (TC-A5).

The suites are run with `python -m pytest tests` in `analysis-backend` and `npm test` in `backend` (`ANALYSIS_LIVE=1` `npm test` also runs TC-A5). Table 5.3 lists the results. All 18 automated cases passed: 14 `pytest` tests covering TC-P1 to TC-P10 (some cases are run for several series lengths), 13 `Jest` tests covering TC-N1 to TC-N3 and TC-A1 to TC-A4, and the live test TC-A5. To confirm that the Node.js tests detect faults rather than pass by default, two defects were deliberately introduced into the controller (a forecast horizon one month too short and a wrong data-quality threshold); both were caught by failing tests, and the controller was then restored. The four system test cases (TC-S1 to TC-S4) are manual tests on a phone; Table 5.3 gives their status, and Section 6.4 describes the runs of TC-S1 and TC-S4.

Table 5.3: Test cases for the forecasting component

ID	Description	Expected result	Req.	Result
TC-P1	Forecast a series of 2 values	method = insufficient_data, no points	FR3	Pass
TC-P2	Forecast a series of 3 values	method = moving_average	FR3	Pass

ID	Description	Expected result	Req.	Result
TC-P3	Forecast series of 4 and 23 values	method = holt_linear_damped	FR3	Pass
TC-P4	Forecast series of 24 and 36 values with a seasonal pattern	method = holt_winters_additive_damped	FR3	Pass
TC-P5	Forecast a steeply falling series	No predicted or lower value below 0	FR4	Pass
TC-P6	Series of 3, 12 and 36 values	$\text{lower} \leq \text{predicted} \leq \text{upper}$; range widens as h grows	FR5	Pass
TC-P7	Same series submitted twice	Identical responses	NFR3	Pass
TC-P8	Series containing a zero month	MAPE computed over non-zero months only	FR6	Pass
TC-P9	Series of exactly 24 and of 30 values	Backtest is null at 24; 6-month holdout at 30	FR6	Pass
TC-P10	POST /forecast with a valid and a malformed body	HTTP 200 with a seasonal forecast; HTTP 422 for the malformed body	NFR6	Pass
TC-N1	addMonths across year boundaries (3 cases)	Correct year and month in each case	FR1	Pass
TC-N2	Horizon summary over 3 months	Totals and bounds equal the sum of the monthly values	FR5	Pass
TC-N3	Data-quality level at 0, 2, 3, 23 and 24 months	none, insufficient, limited, limited, good; no model call below 3 months	FR7	Pass
TC-A1	Request without a token	HTTP 401, code NO_TOKEN	NFR1	Pass
TC-A2	Token without a branch	HTTP 400, code BRANCH_REQUIRED	NFR1	Pass
TC-A3	Branch with 36 complete months and a partial month	HTTP 200; 2 model calls with 36 points and horizon 13; current-month projection, 12 forecast months, 4 horizons	FR1, FR2, FR4	Pass
TC-A4	Analysis service unavailable	HTTP 500 with an error message; no crash	NFR6	Pass
TC-A5	Node.js client against the running Python service, 35 months	Seasonal model, 13 points with valid ranges, 6-month backtest, under 2 s	NFR2	Pass

ID	Description	Expected result	Req.	Result
TC-S1	Open Analytics on a phone with 36 seeded months	Forecast section renders cards, chart and accuracy panel	FR8	Pass (iPhone, 4 Oct 2026)
TC-S2	Branch with 2 complete months	Message asking for at least 3 months; no chart	FR7	Covered by automated TC-N3 (Pass); not run on device
TC-S3	Switch between the four horizons	Cards and monthly breakdown update	FR8	Not yet run (manual)
TC-S4	Device offline	Readable error message with retry	FR8	Pass (iPhone, 4 Oct 2026)

Response time was measured against NFR2 by sending 50 consecutive POST /forecast requests with a 35-month seasonal series to the analysis service on the development laptop. The median response time was 39.5 ms, the 95th percentile 40.8 ms and the maximum 45.0 ms, so model fitting uses well under 1% of the 25-second client timeout. The time of a complete forecast request, which also includes the MongoDB aggregation and both model fits, is reported in Section 6.2.

Request validation provides a further layer of verification at run time: the Pydantic ForecastRequest model rejects malformed input before any computation, and the controller rejects requests without a shop or branch before any database query.

5.5 Deployment and operation

The component runs as two processes alongside the existing SmartCost services: the Node.js backend on port 3000 and the Python analysis service on port 8000. Table 5.4 lists the steps to bring it up in a development or demonstration environment.

Table 5.4: Deployment steps

Step	Command or setting
1. Install analysis-service dependencies	<code>python -m venv .venv and pip install -r requirements.txt in analysis-backend</code>
2. Start the analysis service	<code>uvicorn app.main:app --port 8000</code>

Step	Command or setting
3. Configure the backend	ANALYSIS_SERVICE_URL=http://127.0.0.1:8000 and optionally ANALYSIS_SERVICE_TIMEOUT_MS in backend/.env
4. Start the backend	npm run dev in backend
5. Load demonstration data (optional)	npm run seed:forecast -- --months=36 --reset
6. Run the mobile app	npm start in frontend, with EXPO_PUBLIC_BASE_URL set to the backend address

In production, the analysis service is intended to run on the same host or private network as the backend and is not exposed to the internet; only the Node.js backend calls it, and all tenant checks happen there. Because the service is stateless, it can be scaled horizontally by running more uvicorn workers behind the backend.

The main operational risk is availability. The component deliberately has no JavaScript fallback, so if the analysis service is down or exceeds the 25-second timeout, the forecast endpoint returns HTTP 500 and the app shows an error message, while the rest of the POS keeps working. The /health endpoint of the analysis service allows this to be monitored. Caching the last successful forecast per branch is identified as future work in Chapter 9.

SmartCost POS is in live use. Since September 2026, three businesses have used the platform in their daily operation: Juice N Joy, 77 Brownies and Boat Dynasty. At the time of writing they have at most one complete month of history, so the forecast section shows them the insufficient-data message of Table 4.6 (at least three complete months are needed, and 24 for the seasonal model). For the same reason, their data could not yet be used in the evaluation in this thesis (Section 3.2).

The forecast section is part of the standard Analytics screen and is not behind a feature switch, so it is shown to every branch on the platform, including these three businesses, as soon as they open Analytics.

6 RESULTS AND VALIDATION

This chapter reports the results of the evaluation designed in Chapter 3. The forecasting experiment produced 267,468 scored forecasts across the three datasets. Results are reported here without interpretation; Chapter 7 discusses what they mean.

6.1 Research results

6.1.1 Accuracy by length of history (RQ1)

Table 6.1 and Figure 6.1 show the error of forecasts one to three months ahead, grouped by how many months of history were available at the forecast origin. With 24 or more months, when the ladder uses Holt-Winters, its sales MAPE was 4.3%, against 7.4% for the seasonal naive; for costs the figures were 3.8% and 6.9%. With less history the ladder was less accurate than simpler methods. With 3–5 months its sales MAPE was 16.1% against 9.1% for the 3-month moving average, and with 12–23 months it was 9.8–10.7% against 6.6–6.7% for the seasonal naive.

Table 6.1: Sales MAPE (%) one to three months ahead by months of history, dataset S1

Method	3–5 months	6–11 months	12–17 months	18–23 months	24–35 months
Production ladder	16.1	13.0	9.8	10.7	4.3
Revised ladder (exploratory)	9.1	10.8	6.6	6.7	4.3
Seasonal naive	–	–	6.6	6.7	7.4
3-month moving average	9.1	10.8	10.3	11.5	11.2
Naive (last month)	14.7	15.8	15.9	16.6	16.2
Linear trend	22.4	13.0	10.3	10.9	10.2
Damped Holt only	16.1	13.0	9.8	10.7	10.1

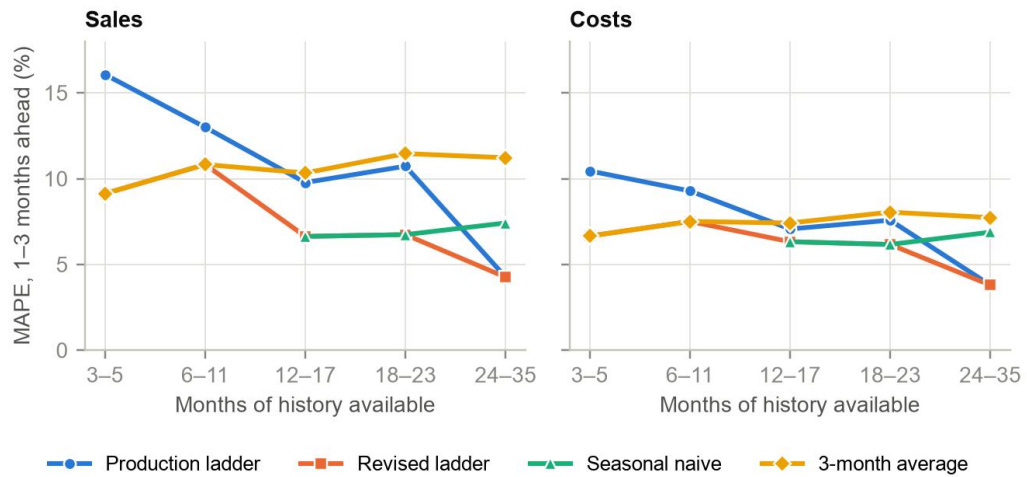


Figure 6.1: Forecast error one to three months ahead by months of history, dataset S1 (the revised ladder coincides with the 3-month average below 12 months, the seasonal naive at 12–23 months and the production ladder from 24 months)

Table 6.2 tests these differences on paired folds for forecasts one month ahead. From 24 months the ladder was more accurate than the seasonal naive in 81.9% of sales folds ($p < 0.001$). In the 12–23-month band the seasonal naive was more accurate in 60.3% of sales folds ($p < 0.001$). The same pattern held for costs and in the noisier dataset S2, with smaller margins.

Table 6.2: Paired comparison of absolute errors one month ahead (Wilcoxon signed-rank test)

Data	Series	History	Ladder vs	Pairs	Median error ladder (LKR)	Median error other (LKR)	Ladder better (%)	p
S1	Costs	12-23	Seasonal naive	360	81,779	74,122	46.1	0.006
S1	Costs	24-35	Seasonal naive	360	41,090	87,537	81.9	< 0.001
S1	Costs	3-11	3-month moving average	270	88,622	68,729	46.7	0.158
S1	Sales	12-23	Seasonal naive	360	245,272	164,640	39.7	< 0.001
S1	Sales	24-35	Seasonal naive	360	101,714	202,590	81.9	< 0.001
S1	Sales	3-11	3-month moving average	270	268,097	206,903	47.4	0.125
S2	Costs	12-23	Seasonal naive	360	87,589	88,657	47.5	0.160
S2	Costs	24-35	Seasonal naive	360	77,984	100,571	63.6	< 0.001
S2	Costs	3-11	3-month moving average	270	108,030	85,758	50.7	0.077
S2	Sales	12-23	Seasonal naive	360	279,968	251,656	45.3	0.008

Data	Series	History	Ladder vs	Pairs	Median error ladder (LKR)	Median error other (LKR)	Ladder better (%)	p
S2	Sales	24-35	Seasonal naive	360	195,279	252,682	62.2	< 0.001
S2	Sales	3-11	3-month moving average	270	301,287	272,023	48.1	0.037

6.1.2 Accuracy by forecast horizon on common folds (RQ1)

Table 6.3 compares the methods on the folds where all of them produce a forecast (origins with at least 12 months). Over these folds the production ladder and the seasonal naive were close one month ahead (7.5% and 7.0% sales MAPE in S1), because the ladder's advantage from 24 months is offset by its weaker 12–23-month band. The exploratory revised ladder had the lowest error at every horizon in S1, 5.4% one month ahead.

Table 6.3: Sales MAPE (%) by horizon on common folds (origins of 12 months or more)

Method	S1 h=1	S1 h=3	S1 h=6	S1 h=12	S2 h=1	S2 h=3	S3 h=1	S3 h=3
Production ladder	7.5	7.0	8.4	9.3	10.7	10.4	8.5	8.4
Revised ladder (exploratory)	5.4	5.7	6.2	7.2	9.6	9.8	5.2	5.5
Seasonal naive	7.0	7.1	7.1	7.3	10.5	10.4	6.0	6.5
3-month moving average	12.9	9.1	10.3	11.7	14.8	11.1	12.7	8.3
Naive (last month)	15.2	15.0	17.3	7.3	16.8	17.0	15.6	15.4
Linear trend	10.9	9.5	10.7	9.2	12.7	11.5	10.8	9.4
Damped Holt only	10.7	9.3	10.5	9.5	12.5	11.4	10.1	9.3

The cost series showed the same ranking with lower errors, because about half of monthly costs are fixed rent, salaries and utilities. The full cost results and MASE values are in Appendix A.

6.1.3 Accuracy of the totals shown to the owner (RQ1)

The app's prediction cards show the total for the selected horizon. Table 6.4 reports the error of these totals. Here the production ladder was the most accurate method in all three datasets: its 3-month sales total was within 4.4% of the actual on average in S1, against 6.7% for the seasonal naive, and its 3-month cost total within 3.5%.

Table 6.4: Error of the 3- and 12-month totals (MAPE of the total, %), common folds

Data	Series	Ladder 3 m	Seasonal naive 3 m	Revised 3 m	Ladder 12 m	Seasonal naive 12 m	Revised 12 m
S1	Sales	4.38	6.75	4.76	4.17	6.73	6.49
S1	Costs	3.54	6.26	4.49	3.55	6.23	6.02
S2	Sales	6.45	7.54	6.64	5.55	6.96	6.79
S2	Costs	4.75	6.42	5.35	4.50	6.25	6.04
S3	Sales	4.94	6.13	5.24	5.35	6.38	6.38
S3	Costs	3.96	6.46	5.50	5.55	6.65	6.65

6.1.4 Coverage of the prediction ranges (RQ2)

Figure 6.2 and Table 6.5 show how often the actual value fell inside the 95% range. One month ahead, coverage was 81.4% for sales and 82.5% for costs, below the nominal 95%. Three months ahead it was close to nominal (95.3% for sales), and at six and twelve months it rose to 97.3% and 99.1%. Ranges were much wider before the seasonal model was available: twelve months ahead the mean width was 151.7% of the actual value with fewer than 24 months of history, against 41.9% with 24 months or more.

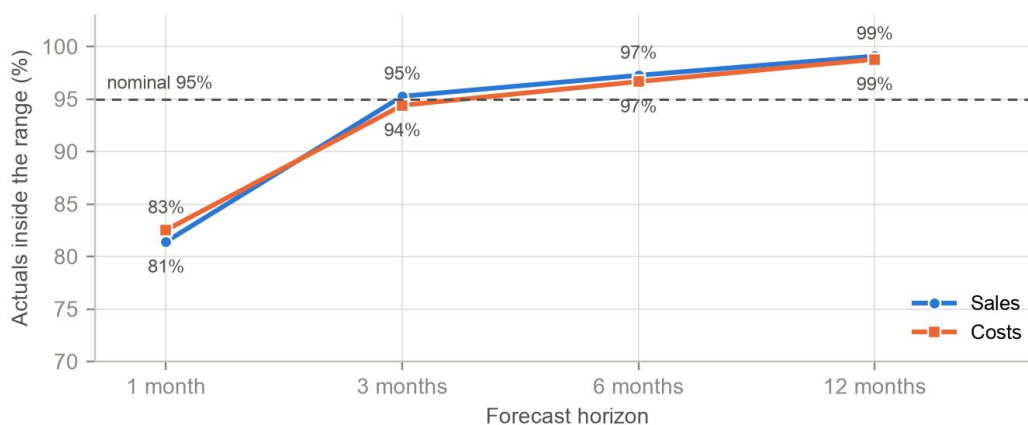


Figure 6.2: Share of actual values inside the 95% prediction range by horizon, dataset S1

Table 6.5: Coverage (%) and mean width (% of actual) of the 95% ranges, dataset S1

Horizon	Sales coverage	Costs coverage	Sales coverage, < 24 months	Sales coverage, ≥ 24 months	Sales width, < 24 months	Sales width, ≥ 24 months
1 month	81.4	82.5	84.6	75.8	47.6	13.1

Horizon	Sales coverage	Costs coverage	Sales coverage, < 24 months	Sales coverage, ≥ 24 months	Sales width, < 24 months	Sales width, ≥ 24 months
3 months	95.3	94.4	96.3	93.0	80.6	22.2
6 months	97.3	96.7	96.8	98.6	112.1	30.7
12 months	99.1	98.8	99.2	96.7	151.7	41.9

6.1.5 Worked example on system-produced data

Figure 6.3 shows a single holdout on dataset S3. Trained on the 24 months from March 2024 to February 2026, the component selected Holt-Winters with $\alpha = 0.25$, $\beta = 0.0$, $\gamma = 0.0$ and $\phi = 0.99$, and forecast March to July 2026 with a MAPE of 2.35%. 4 of the 5 actual values fell inside the 95% range. The forecast reproduced the April peak of the generator's New Year multiplier (1.18) and the dip in June (0.92).

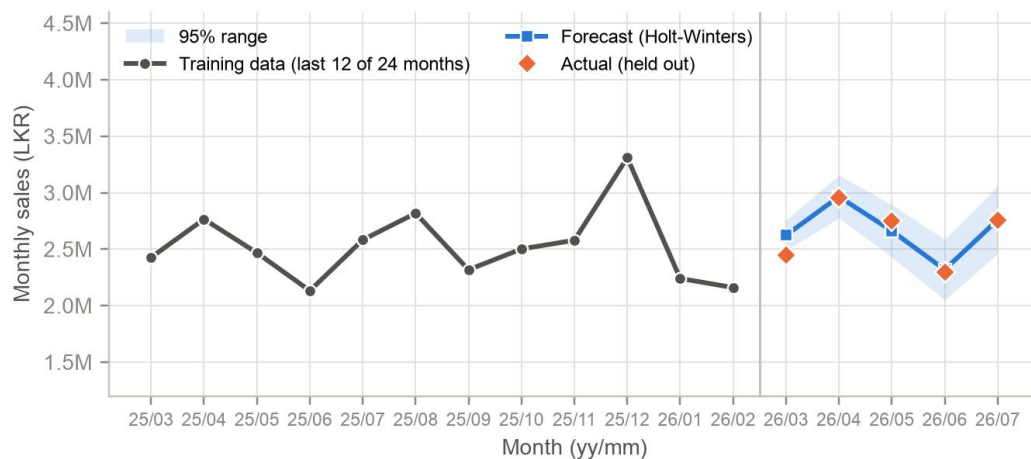


Figure 6.3: Holdout forecast for March to July 2026 on the development branch history (S3)

6.1.6 Robustness to unrecorded months (dataset S4)

In dataset S4 the last two complete months were a half-recorded month (August 2026, LKR 1,564,200) and a month with only three test orders (September 2026, LKR 5,150). With this history, the component forecast zero sales and zero costs for every future month, and its sales backtest reported a MAPE of 8,023.42%. With the history trimmed to July 2026, the same code forecast LKR 2.97 million for August and reported a backtest MAPE of 2.35% (Figure 6.4).

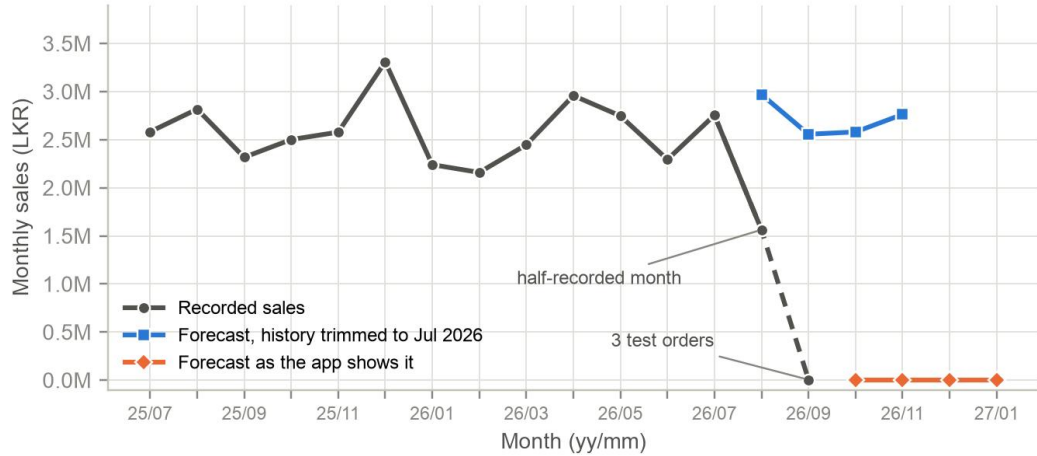


Figure 6.4: Effect of two unrecorded months on the forecast (S4) compared with the history trimmed to July 2026

6.2 Product and technical results

- **Automated tests:** all 18 automated test cases passed (Table 5.3): 14 pytest tests for the forecasting engine and API, 13 Jest tests for the Node.js aggregation logic, controller and API behaviour, and the live contract test. Two deliberately introduced controller defects were both detected.
- **Response time:** the analysis service answered 50 consecutive requests for a 35-month series in a median of 39.5 ms (95th percentile 40.8 ms, maximum 45.0 ms), well within NFR2.
- **Complete request:** for the development branch (35 complete months), the MongoDB Atlas aggregation followed by the two model calls, run as the controller runs them but without authentication, took a median of 328 ms over 20 runs from the development laptop on 9 October 2026 (95th percentile 415 ms, maximum 430 ms), of which 186 ms was the aggregation and 142 ms the two fits. With one analysis-service worker, the two fits took as long when requested in parallel as one after the other (141 ms and 140 ms), because fitting is CPU-bound. The complete request is well within the 2-second target of NFR2.
- **Model selection in practice:** across the 990 folds per series type in S1, the ladder used the moving average 30 times, damped Holt 600 times and Holt-Winters 360 times, matching the thresholds in Figure 4.3.

- **Numerical stability:** statsmodels reported that its optimiser did not fully converge for a small number of fits. These forecasts were kept exactly as the production code produced them, so they are included in all results.
- **User interface:** the Forecast section rendered on a physical iPhone against the backend and analysis service running on the development machine (Figures 4.4 and 5.3), showing the horizon selector, model name, data source and history length.

6.3 User or stakeholder validation

The user study designed in Section 3.3, with the instruments in Appendix A, had not been carried out when this thesis was submitted, so RQ3 is not answered by this thesis (Table 7.1). The three businesses that began using SmartCost in September 2026 (Section 5.5) do not yet have enough history to see a forecast of their own, so the study, when it is run, will use the demonstration branch with 36 months of seeded data, as planned in Section 3.3.

6.4 Integration results

- The forecasting router and the routers of the three other components run in one FastAPI application and share one Node.js client (Section 5.3).
- On a physical device, the complete chain worked: the mobile app authenticated, the backend aggregated 31 months of branch history from MongoDB, the Python service fitted Holt-Winters models, and the app rendered the result with the label "Python · statsmodels".
- The same end-to-end run exposed the unrecorded-month behaviour quantified in Section 6.1.6, which arose from development test data rather than a defect in the integration.
- After the development branch was reseeded with 36 complete months (October 2023 to September 2026), the end-to-end run on an iPhone on 4 October 2026 showed the forecast with the Holt-Winters model, 35 months of history used (the 36-month window includes the month in progress) and backtest MAPEs of 4.99% for sales and 4.7% for costs (Figures 4.5 to 4.7). Test case TC-S1 passed.

- The run revealed two presentation defects in ForecastSection.tsx, neither of which affects the forecast values. First, the past cost points are drawn in the sales colour, because the dot-colour function does not distinguish the two series, although the legend shows costs in red. Second, the chart joins September directly to November: the month in progress is shown in its own card rather than on the chart, but the axis spaces the two months as if they were consecutive. The app's floating settings button also overlaps the right-hand edge of the cards and of the model accuracy panel.
- With the phone in airplane mode, the forecast section replaced its content with a readable message and a Try again button (Figure 6.5), and the rest of the Analytics screen reported the same cause; once the connection was restored, the forecast loaded normally. Test case TC-S4 passed.

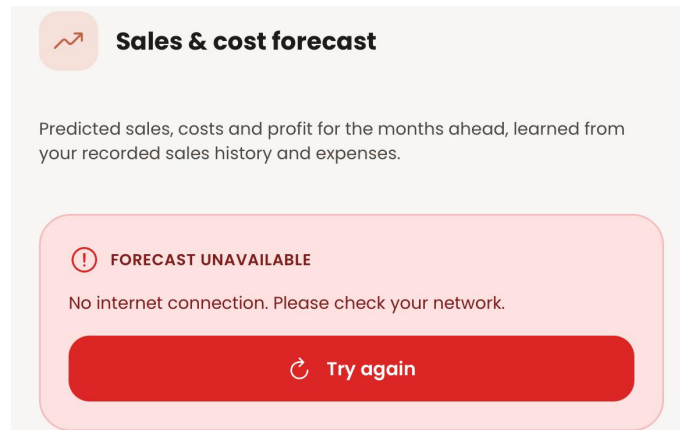


Figure 6.5: Forecast section with no network connection (TC-S4)

- The merges of 31 August 2026 caused two build defects in the mobile app, both outside the forecasting component and fixed the same day (Section 5.3). After the merges, all 18 automated tests of the forecasting component passed (Section 5.4).

6.5 Summary of findings

Table 6.6 summarises the findings against the research questions and objectives.

Table 6.6: Summary of findings against the research questions and objectives

Item	Result	Status
------	--------	--------

Item	Result	Status
RQ1 Accuracy	With 24+ months the ladder halves the seasonal naive's error (4.3% vs 7.4% sales MAPE). With less history it is less accurate than simple rules; horizon totals are the most accurate of all methods.	Partly supported
RQ2 Ranges	Coverage 81.4% at 1 month (too narrow), about 95% at 3 months, 97–99% at 6–12 months (too wide).	Partly supported
RQ3 Usefulness	User study not yet conducted.	Pending
O1 Data pipeline	Implemented and tested (TC-N1, TC-A3).	Met
O2 Model ladder	Implemented; evaluation identifies better thresholds.	Met
O3 Forecast outputs	Implemented and tested (TC-P5, TC-P6, TC-N2, TC-N3).	Met
O4 Evaluation	267,468 scored forecasts, three datasets, six comparison methods.	Met
O5 Integration and validation	Integrated and tested end to end; user validation pending.	Partly met

7 DISCUSSION

This chapter interprets the results of Chapter 6, compares them with the literature and existing products, states the contributions, and discusses the limitations and implications of the work.

7.1 Interpretation of findings

The seasonal rung works. Once 24 months were available, Holt-Winters roughly halved the error of the seasonal naive. The fitted parameters in Section 6.1.5 explain why: with $\gamma = 0$ and $\beta = 0$ the model estimated a fixed seasonal pattern and a stable trend, which matches the generator, and it combined them with a smoothed level that filtered out month-to-month noise. The seasonal naive copies a single past year, noise included.

The lower rungs ignore seasonality for too long. Between 12 and 23 months, damped Holt has no seasonal component, so it reads the seasonal rise towards April or December as a trend and extrapolates it. The seasonal naive, which simply repeats last year, was more accurate in this band. Below 12 months, the damped trend reacted to noise in very short series, and a plain three-month average was more accurate. The thresholds of the ladder were therefore right in principle, a simpler model when there is less data, but the rungs below 24 months were not the best choices. The revised ladder confirms this: using the simplest adequate method at each stage lowered the error at every horizon on the common folds.

Totals are more forgiving than single months. The production ladder gave the most accurate 3- and 12-month totals even where its monthly forecasts were weaker. Without seasonality, damped Holt forecasts close to the average level, so its overestimates of slow months and underestimates of peak months cancel out over a horizon. Because the owner reads totals first on the prediction cards, the practical accuracy of the product is better than its month-by-month error suggests.

The $\sqrt{h}$ range is miscalibrated in both directions. One month ahead, the range uses the spread of the in-sample residuals, which is smaller than the true out-of-sample error because the model was fitted to those same months, so coverage falls short of

95%. At longer horizons, multiplying by $\sqrt{h}$ assumes that errors accumulate like a random walk, but a damped, seasonal series reverts to its pattern, so the ranges grow faster than the errors and become too wide to be informative before 24 months of history.

Missing data is indistinguishable from a collapse. The component fills months without records with zero, which is correct for a closed month but wrong for a month in which the POS was not used. Two such months at the end of the history were enough to drive every forecast to zero.

7.2 Comparison with literature baselines and existing solutions

The results are consistent with the forecasting literature. The strong performance of simple benchmarks below 24 months mirrors the finding of the M4 and M5 competitions that simple statistical methods are hard to beat [3], [4], and the seasonal naive is recommended as a benchmark for seasonal data for exactly this reason [2]. The value of damping is in line with the evidence that damped trends are robust for business series [9], [10]. The miscalibration of the ranges is consistent with the fact that the $\sqrt{h}$ multiplier is exact only for the naive method [2], and it supports evaluating ranges by coverage and width together rather than assuming that a nominal 95% range is reliable.

The accuracy ordering also matches restaurant studies with short histories. Hossain and Parvin found that, with only 15 months of data, simple exponential smoothing outperformed XGBoost, and that all methods had item-level errors above 40% MAPE [13]; in this study, too, the most complex rung was beneficial only once enough history was available. Studies that reported strong machine-learning results used daily data, many series or external variables [4], [5], none of which a single small branch can supply at monthly granularity.

Compared with existing POS products (Section 2.4), the component provides a forward-looking forecast of sales, costs and profit with a stated range, an accuracy figure and an explanation of the data behind it, built into the POS the owner already uses. The closest commercial feature, Toast's sales projections, also shows a range, but it forecasts daily sales up to 28 days ahead and does not forecast costs or profit

[21], [22]. Its accuracy figures are not directly comparable with published restaurant studies, which use different data, granularity and horizons; the comparison that this study can make rigorously is the one against baselines on the same data.

7.3 Research contribution

1. **Evidence on data-size-aware forecasting for young businesses.** The study evaluates a model ladder across the whole range of history lengths a new branch passes through, rather than only on long series, and shows at which stage each method is preferable.
2. **An improved ladder.** The results identify a better rule for the early stages (moving average below 12 months, seasonal naive from 12 to 23 months), which reduced one-month sales error on the common folds of S1 from 7.5% to 5.4%.
3. **Empirical calibration of a simple prediction range.** The study measures how the widely used residual-based range behaves on short monthly business series, showing under-coverage at one month and over-wide ranges at long horizons.
4. **A documented failure mode.** The study identifies and quantifies the effect of unrecorded trailing months on an automatic forecast, a problem specific to forecasting from operational POS data.

7.4 Product contribution

- A working forecasting feature in SmartCost POS, available to every branch automatically from its third month of data.
- A Node.js–Python analysis architecture, tested and documented, that the three other research components reuse.
- Automated test suites and a reproducible evaluation harness that can measure any future change to the models.

7.5 Limitations and threats to validity

Table 7.1 lists the main threats to the validity of the results and how each was mitigated.

Table 7.1: Threats to validity

Type	Threat	Mitigation or consequence
------	--------	---------------------------

Type	Threat	Mitigation or consequence
External	No data from the businesses that use SmartCost in live operation (live only since September 2026); S1–S3 come from one generator with one menu and one seasonal profile.	Thirty branches with rotating start months, a noisier scenario (S2), and system-produced data (S3). Results must be confirmed on real data.
Construct	The generator's seasonality is a fixed multiplicative pattern, which suits Holt-Winters.	Reported as a likely reason for the strong 24+ results; real seasonality will vary from year to year.
Internal	The revised ladder was designed from the same data on which it was evaluated.	Labelled exploratory; it needs confirmation on new data.
Statistical conclusion	Overlapping rolling origins are not independent; S3 gives few folds.	p-values read alongside win shares; conclusions drawn mainly from S1 and S2.
Design	No external drivers; profit errors combine sales and cost errors; empty months treated as zero; no fallback if the analysis service is down.	Listed as future work in Chapter 9.
User study	Not yet conducted.	RQ3 remains open.

7.6 Implications

- **For restaurant owners:** a forecast of totals over the next quarter is already reliable to within about 5% once two years of data exist, and the ranges are a reasonable guide from three months ahead. One-month ranges should be read as optimistic.
- **For POS vendors:** useful forecasting does not require complex machine learning; a few well-chosen statistical models, selected by data size and evaluated against simple baselines, are enough. The bigger practical risk is incomplete data, which must be detected before forecasting.
- **For researchers:** forecasting methods for small businesses should be evaluated across history lengths, because the best method changes as a business grows, and prediction ranges should be checked for coverage rather than assumed.

8 COMMERCIALISATION AND SUSTAINABILITY

This chapter considers the forecasting component as part of a commercial product: who it serves, how it compares with competing products, how it is paid for and operated, the intellectual property and legal obligations involved, and its contribution to the Sustainable Development Goals.

8.1 Value proposition and target users

The target market is small and medium food-service businesses in Sri Lanka: restaurants, cafés, bakeries and canteens, including those with more than one branch. According to the Economic Census 2013/14, Sri Lanka had 1,019,681 non-agricultural establishments, of which 99.8% were micro, small or medium-sized and employed 75.1% of the persons engaged. The accommodation and food service sector had 90,400 establishments, 8.9% of the total, and food and beverage service activities accounted for 83% of these, roughly 75,000 establishments [28]. These businesses are also vulnerable: a survey by the Department of Census and Statistics found that by May 2023, during the economic crisis, 15.7% of the micro, small and medium enterprises active at the end of 2018 had closed permanently [29]. Better planning of purchases and cash is one way to reduce that risk.

The users and their value are:

- **Owners:** an estimate of next quarter's sales, costs and profit, with a range and a statement of how reliable it is, without hiring an analyst or exporting data to a spreadsheet.
- **Branch managers:** a monthly view of expected demand for each branch to plan purchasing and staffing.
- **Accountants or advisers:** a consistent, documented projection to support budgeting and loan applications.

Value proposition: *"Know next month before it happens. SmartCost forecasts your sales, costs and profit automatically from the sales you already record, and tells you how far to trust each number."*

8.2 Market and competitive position

SmartCost has been in live use at three businesses since September 2026 (Section 5.5). Table 8.1 compares SmartCost with the products reviewed in Section 2.4 on published price and forecasting scope. Prices are shown in the currency each vendor publishes and are not converted, because exchange rates and billing terms differ.

Table 8.1: Price and forecasting scope of competing POS products (vendor pages, 4 October 2026)

Product	Published price	Forecasting scope
Square for Restaurants	Free plan USD 0 per month plus 2.6% + 15¢ per in-person transaction [20]	No native sales forecast documented
Toast	Not available (pricing page inaccessible)	Daily sales, 7–28 days, with range
Lightspeed Restaurant	USD 69–399 per month, Starter to Premium [23]	Benchmarks and trends; no sales forecast documented
possystem.lk	Rs. 59,000 desktop; Rs. 39,000 cloud plus Rs. 20,000 per year for the mobile dashboard [24]	Daily "AI forecasting" claimed
TILLMAX	Quotation only [25]	None documented
SmartCost POS	LKR 2,900 per month, or LKR 8,000 / 15,600 / 30,600 for 3 / 6 / 12 months; 14-day trial	Monthly sales, costs and profit, 1–12 months, with range and accuracy

SmartCost's position is a low-cost, locally priced POS whose forecasting covers the whole business (sales, costs and profit) rather than labour scheduling alone. Its main competitive risks are that international vendors extend their forecasting to monthly financials, and that local competitors add similar features. Two aspects are harder to copy quickly: the evaluated model ladder, which keeps the forecast sensible for a business in its first two years, and the integration with SmartCost's cost and expense records, which are needed for a cost and profit forecast.

8.3 Business or sustainability model

SmartCost is sold as a subscription, with optional charges for additional users (LKR 290 per user per month), the web portal (LKR 2,990 per month) and SMS packages. The forecasting component is not priced separately: it is included for every subscribed branch as a differentiating feature that supports conversion at the end of the 14-day trial and retention afterwards. Because the forecast becomes more

accurate as history accumulates, its value to the owner grows the longer the business stays on the platform.

The operating cost of the feature is low. Each forecast is computed from at most 36 monthly values per series; the median response time of the analysis service measured in Section 5.4 was under 40 ms, and the analysis service needs no GPU or model training infrastructure. The same analysis service also hosts the other three research components. The technical sustainability of the feature rests on the automated test suites and the evaluation harness, which allow any change to the models to be re-evaluated against the same baselines before release.

A possible future premium tier could add daily forecasts, product-level purchasing suggestions (combined with component 3) or forecasts that use holidays and promotions. Its price would need to be set after the user study has measured how much owners value the basic forecast.

8.4 Scalability, intellectual property and compliance

8.4.1 Scalability

The analysis service is stateless: each request carries the series to be forecast, so additional instances can be added behind a load balancer without coordination. The database query reads at most 36 months per branch and groups them in MongoDB, so its cost does not grow with the age of the business. Because the forecast changes only when a month closes or past records are edited, results could be cached per branch and month, which would reduce the number of model fits to roughly one per branch per month. The 25-second client timeout and the fallback chain of models (Section 4.4) prevent one slow or unusual series from affecting other users.

8.4.2 Intellectual property

The component is built on open-source libraries with permissive licences that allow commercial use, provided their copyright notices are retained (Table 8.2). No library with a copyleft licence is used in the forecasting path. The forecasting design, the model ladder and the evaluation harness are original work of the project. Ownership and any registration of the project's intellectual property are governed by the university's regulations for student research projects.

Table 8.2: Licences of the main libraries used by the component

Library	Used for	Licence
statsmodels, NumPy, pandas, SciPy	Exponential smoothing and numerical work	BSD 3-Clause
FastAPI, Pydantic	Analysis service API and validation	MIT
Uvicorn	ASGI server for the analysis service	BSD 3-Clause
Express, Mongoose, axios, jsonwebtoken	Backend API, data access, service client, authentication	MIT
React Native, Expo, react-native-chart-kit	Mobile app and forecast chart	MIT

8.4.3 Legal and regulatory compliance

Sri Lanka's Personal Data Protection Act No. 9 of 2022 [30], amended by Act No. 22 of 2025, regulates the processing of personal data. Its provisions come into operation in stages, by order of the Minister published in the Gazette, so the obligations in force should be confirmed against current Gazette orders before commercial deployment. The forecasting component itself processes only monthly monetary totals per branch: the aggregation reads the amount and date of each order and expense and returns no customer or employee data, which follows the principle of data minimisation. Access is limited to authenticated users of the same shop and branch through JSON Web Tokens, addressing broken object-level authorisation, the first risk in the OWASP API Security Top 10. Data from real businesses used for future evaluation will require the owner's consent and anonymisation, as described in Section 3.6.

8.5 Contribution to the Sustainable Development Goals

Table 8.3 maps the component to targets of the United Nations 2030 Agenda for Sustainable Development. The contributions are indirect: the component provides information, and the benefit depends on how owners use it.

Table 8.3: Contribution to the Sustainable Development Goals

SDG target	Contribution of the component	Strength
8.3 Support the growth of micro, small and medium-sized enterprises	Gives small restaurants a forecasting capability that was previously available only to larger businesses with analysts, helping them plan cash and staffing	Direct, modest

SDG target	Contribution of the component	Strength
9.3 Increase the access of small-scale enterprises to financial services and their integration into markets	A documented projection of sales, costs and profit can support budgeting and loan applications	Indirect
12.3 Halve per capita food waste at the retail and consumer levels	Anticipating slow and busy months helps owners buy ingredients closer to demand; item-level purchasing support would strengthen this	Indirect

Measuring these effects, for example the change in ingredient waste or cash shortfalls after adoption, would require a longitudinal study with operating restaurants and is proposed as future work.

9 CONCLUSION AND FUTURE DIRECTIONS

This chapter summarises the work, assesses how far each objective was achieved, states the individual contributions and proposes directions for future work.

9.1 Conclusion

This thesis set out to give small restaurants an automatic, trustworthy forecast of their sales, costs and profit inside the POS they already use. A forecasting component was designed and built for SmartCost POS: it aggregates each branch's orders and expenses into monthly series, selects a moving average, damped Holt or damped Holt-Winters model according to the history available, and shows 1- to 12-month forecasts with prediction ranges, measured accuracy and a plain-language data-quality message in the mobile app.

The evaluation showed that the component is accurate once two years of history exist, with a sales MAPE of 4.3% one to three months ahead, and that, from 12 months of history, the 3-month totals shown to owners are within about 4.4% for sales. It also showed that the early rungs of the ladder and the prediction ranges can be improved, and that unrecorded months must be detected before forecasting. The component is integrated and tested end to end; its validation with owners remains to be completed.

9.2 Achievement of objectives

Table 9.1 summarises how far each objective was achieved.

Table 9.1: Achievement of objectives

Objective	How it was achieved	Evidence	Status
O1 Data pipeline	MongoDB monthly aggregation in Asia/Colombo time with zero-filling and partial-month exclusion	Sections 4.4.1, 5.2.1; TC-N1, TC-A3	Met
O2 Model ladder	Three-rung ladder with fallback chain	Section 4.3.2, Figure 4.3; TC-P1–P4	Met
O3 Forecast outputs	Horizons, ranges, accuracy, data quality	Sections 4.4, 5.2.2; TC-P5, TC-P6, TC-N2, TC-N3	Met
O4 Evaluation	Rolling-origin backtest on three datasets against six methods	Section 6.1	Met

Objective	How it was achieved	Evidence	Status
O5 Integration and validation	Integrated into backend and app; 18 automated tests; user study pending	Sections 5.3, 5.4, 6.2–6.4	Partly met

9.3 Individual research and product contributions

- Designed and implemented the forecasting component end to end: aggregation, orchestration, the Python forecasting engine and API, and the mobile interface.
- Created the Node.js–Python analysis architecture reused by the three other research components.
- Built the synthetic data generator, the automated test suites and a reproducible evaluation harness.
- Produced evidence on how forecast accuracy and range calibration change with the length of a small business's history, an improved model ladder, and a quantified failure mode for unrecorded months.

9.4 Future directions

1. **Adopt the revised ladder** (moving average below 12 months, seasonal naive for 12–23 months) after confirming it on new data.
2. **Add a data-completeness guard** that compares recent order counts with the branch's normal level, trains only on months that look fully recorded, and tells the owner which months were excluded.
3. **Calibrate the prediction ranges** from the empirical distribution of backtest errors at each horizon, or with model-based simulation, and re-check coverage.
4. **Validate with owners and real data:** run the user study and evaluate the component on the history of operating SmartCost restaurants.
5. **Add explanatory variables** such as Sri Lankan public holidays and promotions, and evaluate whether they reduce error.
6. **Extend granularity and scope:** weekly forecasts for staffing, and links to the product demand component for purchase planning.
7. **Improve resilience:** cache the last successful forecast per branch and serve it if the analysis service is unavailable.

REFERENCES

- [1] A. Lasek, N. Cercone, and J. Saunders, "Restaurant sales and customer demand forecasting: Literature survey and categorization of methods," in *Smart City 360°* (LNICST, vol. 166), A. Leon-Garcia *et al.*, Eds. Cham, Switzerland: Springer, 2016, pp. 479–491, doi: 10.1007/978-3-319-33681-7_40.
- [2] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 3rd ed. Melbourne, Australia: OTexts, 2021. [Online]. Available: <https://otexts.com/fpp3/>
- [3] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: 100,000 time series and 61 forecasting methods," *Int. J. Forecast.*, vol. 36, no. 1, pp. 54–74, 2020, doi: 10.1016/j.ijforecast.2019.04.014.
- [4] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "M5 accuracy competition: Results, findings, and conclusions," *Int. J. Forecast.*, vol. 38, no. 4, pp. 1346–1364, 2022, doi: 10.1016/j.ijforecast.2021.11.013.
- [5] A. Schmidt, M. W. U. Kabir, and M. T. Hoque, "Machine learning based restaurant sales forecasting," *Mach. Learn. Knowl. Extr.*, vol. 4, no. 1, pp. 105–130, 2022, doi: 10.3390/make4010006.
- [6] K. Posch, C. Truden, P. Hungerländer, and J. Pilz, "A Bayesian approach for predicting food and beverage sales in staff canteens and restaurants," *Int. J. Forecast.*, vol. 38, no. 1, pp. 321–338, 2022, doi: 10.1016/j.ijforecast.2021.06.001.
- [7] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *Int. J. Forecast.*, vol. 20, no. 1, pp. 5–10, 2004, doi: 10.1016/j.ijforecast.2003.09.015. (Reprint of ONR Research Memorandum No. 52, Carnegie Institute of Technology, 1957.)
- [8] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," *Manage. Sci.*, vol. 6, no. 3, pp. 324–342, 1960, doi: 10.1287/mnsc.6.3.324.
- [9] E. S. Gardner Jr. and E. McKenzie, "Forecasting trends in time series," *Manage. Sci.*, vol. 31, no. 10, pp. 1237–1246, 1985, doi: 10.1287/mnsc.31.10.1237.
- [10] E. S. Gardner Jr., "Exponential smoothing: The state of the art—Part II," *Int. J. Forecast.*, vol. 22, no. 4, pp. 637–666, 2006, doi: 10.1016/j.ijforecast.2006.03.005.
- [11] S. J. Taylor and B. Letham, "Forecasting at scale," *Amer. Statistician*, vol. 72, no. 1, pp. 37–45, 2018, doi: 10.1080/00031305.2017.1380080.
- [12] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward," *PLoS ONE*, vol. 13, no. 3, Art. no. e0194889, 2018, doi: 10.1371/journal.pone.0194889.

- [13] M. S. Hossain and F. Parvin, "A comparative study of various statistical and machine learning models for predicting restaurant demand in Bangladesh," *PLoS ONE*, vol. 20, no. 6, Art. no. e0325449, 2025, doi: 10.1371/journal.pone.0325449.
- [14] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: An analysis and review," *Int. J. Forecast.*, vol. 16, no. 4, pp. 437–450, 2000, doi: 10.1016/S0169-2070(00)00065-0.
- [15] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *Int. J. Forecast.*, vol. 22, no. 4, pp. 679–688, 2006, doi: 10.1016/j.ijforecast.2006.03.001.
- [16] A. M. van der Bles, S. van der Linden, A. L. J. Freeman, and D. J. Spiegelhalter, "The effects of communicating uncertainty on public trust in facts and numbers," *Proc. Natl. Acad. Sci. USA*, vol. 117, no. 14, pp. 7672–7683, 2020, doi: 10.1073/pnas.1913678117.
- [17] F. D. Davis, "Perceived usefulness, perceived ease of use, and user acceptance of information technology," *MIS Quarterly*, vol. 13, no. 3, pp. 319–340, 1989, doi: 10.2307/249008.
- [18] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation in Industry*, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I. L. McClelland, Eds. London, U.K.: Taylor & Francis, 1996, pp. 189–194.
- [19] R. J. Hyndman and A. V. Kostenko, "Minimum sample size requirements for seasonal forecasting models," *Foresight*, no. 6, pp. 12–15, 2007.
- [20] Square, "Square for Restaurants pricing." Accessed: Oct. 4, 2026. [Online]. Available: <https://squareup.com/us/en/point-of-sale/restaurants/pricing>
- [21] Toast, "Toast Benchmarking: Advanced sales projections," Toast Central. Accessed: Oct. 4, 2026. [Online]. Available: <https://support.toasttab.com/en/article/Toast-Benchmarking-Advanced-Sales-Projections>
- [22] Sling, "Projected sales forecasting for Toast integration," Sling Help Center. Accessed: Oct. 4, 2026. [Online]. Available: <http://support.getsling.com/en/articles/8614345-projected-sales-forecasting-for-toast-integration>
- [23] Lightspeed Commerce, "Restaurant POS pricing." Accessed: Oct. 4, 2026. [Online]. Available: <https://www.lightspeedhq.com/pos/restaurant/pricing/>
- [24] Applantics (Pvt) Ltd, "POS System Sri Lanka," possystem.lk. Accessed: Oct. 4, 2026. [Online]. Available: <https://possystem.lk/>
- [25] TILLMAX, "TILLMAX POS," tillmax.lk. Accessed: Oct. 4, 2026. [Online]. Available: <https://www.tillmax.lk/>

- [26] S. Seabold and J. Perktold, "statsmodels: Econometric and statistical modeling with Python," in *Proc. 9th Python in Science Conf. (SciPy 2010)*, 2010, pp. 92–96, doi: 10.25080/Majora-92bf1922-011.
- [27] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–106, 2004, doi: 10.2307/25148625.
- [28] Department of Census and Statistics, Sri Lanka, "Economic Census 2013/2014: Listing stage report." Accessed: Oct. 4, 2026. [Online]. Available: https://www.statistics.gov.lk/Resource/en/Industry/Other_Tables_Reports/EconomicCensus2013-2014_ListingStageReport.pdf
- [29] Department of Census and Statistics, Sri Lanka, "Impact of the economic crisis on MSMEs in Sri Lanka," media release, Feb. 13, 2024. Accessed: Oct. 4, 2026. [Online]. Available: https://www.statistics.gov.lk/qlink/PressReleases/MSMEs_Media%20Release_English
- [30] *Personal Data Protection Act, No. 9 of 2022*, as amended by Act No. 22 of 2025, Parliament of the Democratic Socialist Republic of Sri Lanka, Mar. 2022. [Online]. Available: [https://dpa.gov.lk/acts/Data%20Protection%20Act%20SL%20-%20English%20\(2\).pdf](https://dpa.gov.lk/acts/Data%20Protection%20Act%20SL%20-%20English%20(2).pdf)

GLOSSARY

Backtesting. Evaluating a forecasting method by forecasting past periods from the data that preceded them and comparing the forecasts with what actually happened.

Coverage. The share of actual values that fall inside a prediction range; a 95% range should cover about 95% of actuals.

Damped trend. A trend component whose effect shrinks with each step ahead, so that long-range forecasts flatten instead of growing without limit.

Exponential smoothing. A family of forecasting methods in which forecasts are weighted averages of past observations, with weights that decrease exponentially with age.

Forecast horizon. How many periods ahead a forecast is made; $h = 3$ means three months after the last observed month.

Forecast origin. The last period of data available when a forecast is made.

Holt-Winters method. Exponential smoothing with level, trend and seasonal components.

Model ladder. The rule by which the component chooses a simpler or more complex forecasting model according to the number of months of history.

Multi-tenant. A single software installation serving many separate businesses whose data is kept apart.

Prediction range. An interval within which a future value is expected to fall with a stated probability; also called a prediction interval.

Rolling-origin evaluation. Backtesting repeated from many successive forecast origins.

Seasonal naive method. A benchmark that forecasts each month to be equal to the same month one year earlier.

Seasonality. A pattern that repeats over a fixed period, such as higher restaurant sales every December.

Smoothing parameter. A value between 0 and 1 (α , β , γ) that controls how strongly a model reacts to the latest observation.

APPENDICES

Appendix A Research instruments or extended results

This appendix contains the extended results referred to in Chapter 6 and the instruments for the user study. The complete forecast-level results are in analysis-backend/experiments/results/fold_forecasts.csv.

A.1 Extended forecasting results

Tables A.1 to A.5 extend the results of Section 6.1: costs and dataset S2 by length of history (Tables A.1 and A.2), MASE and costs by horizon (Tables A.3 and A.4), and paired tests three months ahead (Table A.5).

Table A.1: Costs MAPE (%) one to three months ahead by months of history, dataset S1

Method	3–5 months	6–11 months	12–17 months	18–23 months	24–35 months
Production ladder	10.5	9.3	7.1	7.6	3.8
Revised ladder (exploratory)	6.7	7.5	6.3	6.2	3.8
Seasonal naive	–	–	6.3	6.2	6.9
3-month moving average	6.7	7.5	7.4	8.1	7.7
Naive (last month)	9.9	10.9	11.0	11.4	10.9
Linear trend	15.5	9.2	7.4	7.7	7.0
Damped Holt only	10.5	9.3	7.1	7.6	7.0

Table A.2: Sales MAPE (%) one to three months ahead by months of history, dataset S2

Method	3–5 months	6–11 months	12–17 months	18–23 months	24–35 months
Production ladder	15.8	15.7	11.7	12.6	9.0
Revised ladder (exploratory)	11.5	12.8	10.6	10.2	9.0
Seasonal naive	–	–	10.6	10.2	10.5
3-month moving average	11.5	12.8	12.4	13.2	13.3
Naive (last month)	16.1	18.8	17.7	18.5	18.0
Linear trend	23.2	15.1	12.3	12.8	12.1
Damped Holt only	15.8	15.7	11.7	12.6	12.2

Table A.3: MASE by horizon on common folds, dataset S1

Method	Sales h=1	Sales h=3	Sales h=12	Costs h=1	Costs h=3	Costs h=12
Production ladder	1.16	1.12	1.71	1.24	1.20	1.63
Revised ladder (exploratory)	0.85	0.92	1.30	0.94	0.97	1.63
Seasonal naive	1.11	1.14	1.32	1.20	1.21	1.66
3-month moving average	1.98	1.47	2.14	1.80	1.37	2.09
Naive (last month)	2.33	2.42	1.32	1.99	2.12	1.66
Linear trend	1.66	1.48	1.67	1.54	1.39	1.56
Damped Holt only	1.67	1.49	1.75	1.54	1.42	1.64

Table A.4: Costs MAPE (%) by horizon on common folds

Method	S1 h=1	S1 h=3	S1 h=6	S1 h=12	S2 h=1	S2 h=3	S3 h=1	S3 h=3
Production ladder	5.7	5.4	6.4	6.9	7.7	7.4	7.0	6.1
Revised ladder (exploratory)	5.0	5.2	5.7	6.6	7.3	7.4	5.7	5.8
Seasonal naive	6.6	6.6	6.6	6.7	8.1	8.0	6.3	6.7
3-month moving average	8.9	6.5	7.5	8.8	10.1	7.8	9.6	6.6
Naive (last month)	10.3	10.4	12.0	6.7	11.5	11.8	11.5	11.0
Linear trend	7.6	6.7	7.5	6.5	8.7	7.9	8.1	6.9
Damped Holt only	7.5	6.6	7.5	6.9	8.7	8.0	7.9	6.9

Table A.5: Paired comparison of absolute errors three months ahead (Wilcoxon signed-rank test)

Data	Series	History	Ladder vs	Pairs	Median error ladder (LKR)	Median error other (LKR)	Ladder better (%)	p
S1	Costs	12-23	Seasonal naive	360	67,268	74,259	53.9	0.847
S1	Costs	24-35	Seasonal naive	300	40,295	89,130	83.3	< 0.001
S1	Costs	3-11	3-month moving average	270	82,019	53,353	38.9	< 0.001
S1	Sales	12-23	Seasonal naive	360	197,810	167,940	46.1	< 0.001
S1	Sales	24-35	Seasonal naive	300	100,540	209,835	78.7	< 0.001
S1	Sales	3-11	3-month moving average	270	264,943	168,140	31.9	< 0.001

Data	Series	History	Ladder vs	Pairs	Median error ladder (LKR)	Median error other (LKR)	Ladder better (%)	p
S2	Costs	12-23	Seasonal naive	360	76,075	91,590	51.4	0.610
S2	Costs	24-35	Seasonal naive	300	81,085	98,839	63.3	< 0.001
S2	Costs	3-11	3-month moving average	270	85,102	70,512	40.0	< 0.001
S2	Sales	12-23	Seasonal naive	360	231,953	245,167	50.6	0.318
S2	Sales	24-35	Seasonal naive	300	213,679	255,916	59.3	< 0.001
S2	Sales	3-11	3-month moving average	270	247,054	203,065	36.3	< 0.001
S3	Costs	12-23	Seasonal naive	12	73,545	82,129	58.3	0.733
S3	Sales	12-23	Seasonal naive	12	227,341	151,410	33.3	0.064

A.2 User study instruments

Consent statement. "I agree to take part in a study of the SmartCost forecast feature conducted by a SLIIT undergraduate. I understand that my participation is voluntary, that I may stop at any time, that no financial data of my business will be collected, and that my answers will be reported anonymously." Signature and date.

Tasks. (1) Read the expected sales for the next three months and their range. (2) Decide whether costs are expected to rise or fall over the next six months. (3) Explain how reliable the forecast is, using the screen.

System Usability Scale [18], adapted to the forecast feature. Each item is rated from 1 (strongly disagree) to 5 (strongly agree).

1. I think that I would like to use this forecast feature frequently.
2. I found the forecast feature unnecessarily complex.
3. I thought the forecast feature was easy to use.
4. I think that I would need the support of a technical person to be able to use this forecast feature.
5. I found the various functions in this forecast feature were well integrated.
6. I thought there was too much inconsistency in this forecast feature.
7. I would imagine that most people would learn to use this forecast feature very quickly.

8. I found the forecast feature very cumbersome to use.
9. I felt very confident using the forecast feature.
10. I needed to learn a lot of things before I could get going with this forecast feature.

Perceived usefulness (adapted from the technology acceptance model [17]), rated from 1 to 5.

1. Using the forecast would help me plan ingredient and stock purchases.
2. Using the forecast would help me plan staffing.
3. Using the forecast would help me plan cash flow and payments.
4. The range shown with the forecast helps me judge how much to trust it.
5. I would check the forecast at least once a month.

Interview guide.

1. How do you currently estimate next month's sales and costs?
2. In your own words, what does the forecast on this screen tell you?
3. What does the range under each number mean to you?
4. Which decision, if any, would you make differently because of this forecast?
5. What would make you trust the forecast more, or less?
6. What is missing from the screen that you would need?

Appendix B Academic integrity and AI use disclosure

Table B.1 lists the use of generative AI tools in this project, as required by the declaration at the start of this thesis. The student remains responsible for the accuracy, originality, citations, code, data, analyses and conclusions.

Table B.1: Use of generative AI tools

Tool	Purpose	What it did	Verification	Affected
Claude Code (Anthropic), Claude Opus 5.5 model	Thesis planning	Produced a section-by-section writing guide from the template and the project code.	Reviewed by the student.	Structure of all chapters
Same	Drafting text	Drafted Chapters 1–9, the abstract, glossary and appendices from the project source code and the saved evaluation outputs.	All numbers are generated by scripts from the saved results files.	Chapters 1–9, Abstract, Appendices

Tool	Purpose	What it did	Verification	Affected
Same	Figures	Drew Figures 4.1–4.3 and wrote the script that produces Figures 6.1–6.4.	Figures checked against the code and the results files.	Figures 4.1–4.3, 6.1–6.4
Same	Software	Wrote the automated tests, the evaluation and figure scripts, and the timing script used in Section 6.2.	Tests executed (18 pass); two injected defects detected; evaluation re-run from scratch.	tests/, experiments/
Same	Literature	Searched for and checked bibliographic details of the sources cited.	Each reference checked against the publisher or DOI page.	Chapter 2, References