

SmartLife AI: An Intelligent, Inflation-Aware Personal Finance Management System - Design, Implementation, and a Roadmap Toward Predictive Financial Intelligence

Om Kailash Sarjine¹, Sarthak Surendra Posam², Samruddhi Subhash Patil³,
Sarvesh Rajesh Bangad⁴, Prof. Sheetal Shimpikar⁵

^{1, 2, 3, 4} Dept of Information Technology

⁵ Guide, Dept of Information Technology

^{1, 2, 3, 4, 5} Pillai College of Engineering, New Panvel, Navi Mumbai, India

Abstract- *Personal financial management (PFM) applications typically report nominal income, expenses, and balances without indicating how inflation erodes purchasing power or whether a user is on track for long-term goals such as retirement. This paper presents SmartLife AI, a full-stack, inflation-aware personal finance platform built on a React/Next.js frontend and a Python FastAPI backend. The current (Phase 1) implementation — 32 REST endpoints across 19 database entities, validated by 24 automated unit tests — provides financial record management, bank-statement ingestion with rule-based categorization, and four deterministic engines: Inflation, Real Wealth, Life Value, and a Retirement Projection Engine with accumulation-and-drawdown simulation, alongside an FY 2025-26 Indian income-tax optimization engine and a linear-regression expense forecasting baseline. Every worked example in this paper is computed directly from the deployed engines. The paper also specifies the Phase 2 roadmap — comparative ARIMA/LSTM forecasting, a myScheme-aligned government-scheme recommender, a credit-card suitability scorer, and a Gemini-based advisory layer — establishing a reproducible baseline for the completed half of the project and a falsifiable specification for the remainder.*

Keywords- Personal Financial Management, Inflation-Adjusted Wealth, Retirement Planning, Financial Decision Support, Machine Learning, FastAPI, Next.js, Recommendation Systems, Income Tax Optimization, Artificial Intelligence

I. INTRODUCTION

Most digital personal-finance tools available today do a good job of showing what a user has earned and spent, but they stop there— they don't tell the user what that money will actually be worth down the line, or whether their current habits are enough to hit a goal like retirement. That's a real

problem over long time horizons, because inflation quietly eats into savings year after year. Someone who only checks their nominal account balance can feel like they're getting ahead financially when, once you adjust for purchasing power, they're actually standing still or falling behind. SmartLife AI was built to address exactly this blind spot.

SmartLife AI was built to address exactly this blind spot. Unlike a purely conceptual proposal, the system described in this paper has a working implementation: a Next.js/React frontend, a FastAPI backend with SQLAlchemy-backed persistence across nineteen relational entities, and a set of deterministic calculation engines for inflation, real wealth, life value, retirement planning, and Indian income tax, alongside a first-generation machine-learning expense forecaster and a rule-based investment allocation recommender. This paper reports the system as it exists today (Phase 1) and specifies, with equal precision, the machine-learning and recommendation components planned for Phase 2.

The motivation for treating this as an explicit two-phase project is methodological honesty: a system that reports fabricated accuracy figures for models that were never trained is not defensible in a viva or a peer review, whereas a system that clearly separates “what is built and verified” from “what is designed and scheduled” is. Section VI of this paper therefore reports only outputs that were generated by directly executing the project's own calculation modules, rather than assumed or estimated figures. Every table, chart, and worked example in this paper comes directly from calling the deployed code— nothing here is hand-calculated or estimated. Section IV-G describes the automated test suite that independently checks the same engines through continuous integration.

The contributions of this paper are:

- A working, deployed architecture for inflation-aware personal financial management, combining a rule-based transaction pipeline with deterministic financial-mathematics engines, exposed through 32 REST endpoints.
- A monthly-compounding retirement-projection engine that simulates both the accumulation phase (SIP with annual stepup) and the drawdown phase (inflation-indexed withdrawals), computing a required corpus, a projected corpus, and a readiness percentage.
- A tax engine for FY 2025-26 that compares old-and-new-regime liability for Indian taxpayers and points out unused Section 80C/80D deduction headroom.
- A Phase 2 roadmap that we've deliberately made specific and checkable rather than aspirational— it lays out planned work on comparative ML forecasting (linear regression vs. ARIMA vs. LSTM), a government-scheme eligibility recommender tied to the official myScheme platform, a credit-card suitability scorer, and a Gemini-based conversational layer.
- Worked numerical examples and figures generated directly from the deployed codebase, rather than assumed values, backed by a 24-test automated verification suite, to keep every reported number reproducible.

II. RELATED WORK

A. Machine Learning for Financial Forecasting and Planning

Wasserbacher and Spindler [1] reviewed the use of machine learning in financial forecasting and planning and emphasized that accurate prediction of a variable does not automatically translate into an optimal planning decision — forecasts must be embedded in a decision-support layer to be useful. Zhang, Sjarif, and Ibrahim [7] reviewed deeplearning architectures (including LSTM-based models) for financial time-series price forecasting, noting the trade-off between model capacity and the volume of training data required to avoid overfitting. This trade-off directly motivates SmartLife AI's Phase 1 choice of a simple, transparent linear-regression baseline for expense forecasting, with ARIMA/LSTM comparison deferred to Phase 2 once a sufficiently long per-user transaction history is available.

B. Inflation Forecasting and Macroeconomic Modelling

Bareith, Tatay, and Vancsura [2] demonstrated AI-based portfolio-management insights under inflation, and Malhotra and Maloo [3] applied a machine-learning approach to identify drivers of food inflation in India, illustrating the

value of category-specific modelling in a country with substantial regional and commodity-level price variation. Limawan et al. [14] used Random Forest regression for macroeconomic CPI forecasting from GDP, unemployment and related indicators.

These works motivate the Phase 2 inflation-forecasting module described in Section VIII, while the Phase 1 system uses a configurable, user-editable inflation-rate assumption grounded in official Consumer Price Index data published by India's Ministry of Statistics and Programme Implementation [12].

C. AI-Assisted Personal Finance Systems

Mahto et al. [4] proposed an ML-based expense tracker and spending predictor; Harshitha, Kumar, and Rakshitha [5] investigated AI-driven financial insights for personal budget planning; and Jadhav et al. [6] presented an AI-assisted PFM system built on Flask. These systems demonstrate the feasibility of ML-assisted personal-finance tooling but generally treat categorization, forecasting, and recommendation as isolated features rather than components of a single inflation-aware decision pipeline — the gap SmartLife AI targets.

D. Recommendation Systems and Conversational AI

Financial-product and content recommendation has been studied extensively outside the PFM domain. Covington, Adams, and Sargin [15] demonstrated deep neural recommendation at scale for YouTube, and He et al. [16] proposed Neural Collaborative Filtering for combining matrix factorization with neural interaction modelling; both inform the suitability-scoring design intended for SmartLife AI's Phase 2 credit-card recommender (Section VIII-C). For the planned conversational advisory layer, Imran and Almusharraf [17] reviewed Google Gemini as an emerging educational and reasoning tool, and Devlin et al. [18] and Reimers and Gurevych [19] established the transformer-based sentence-encoding techniques (BERT, Sentence-BERT) that underlie modern retrieval-augmented conversational systems — an architecture pattern also used in a related institution project, an AI-personalized educational conversational agent built on Whisper, retrieval-augmented generation, and the Gemini API [20], which the SmartLife AI conversational layer is expected to draw design patterns from (structured system-instruction conditioning, retrieved context, and turn-level dialogue history).

E. Government Scheme Discovery and Regulatory Basis

India's official myScheme platform provides centralized, eligibility-based discovery of government welfare schemes [11]. SmartLife AI's planned scheme-recommendation module (Section VIII-B) is explicitly designed as a discovery and pre-filtering layer that directs users to this authoritative source rather than replacing it. Similarly, the planned credit-card suitability module (Section VIII-C) is designed around the Reserve Bank of India's Master Direction on credit- and debit-card issuance and conduct [13], which governs permissible disclosure and marketing practices for card products in India.

F. Research Gap

Across the reviewed literature, three gaps recur: (i) inflation is rarely connected to individual expense and wealth trajectories in a way a user can act on; (ii) forecasting and recommendation are usually developed as standalone features rather than a single pipeline; and (iii) systems that do integrate multiple features rarely disclose which components are empirically evaluated versus conceptually proposed. SmartLife AI addresses the first two gaps through its deterministic inflation/real-wealth/lifevalue/retirement pipeline and its Phase 2 roadmap, and addresses the third by explicitly separating implemented from proposed components throughout this paper (Fig. 1 and Fig. 5).

G. Existing Systems vs. Our Contribution

Capability	Existing [1][6],[14]	SmartLife AI
Expense tracking & categorization	Yes	Yes, LLMextensible
Inflation as static input only	Yes, mostly	Category-aware math
Nominal wealth reporting only	Yes	Real Wealth Engine
Life-value / opportunity-cost view	Not addressed	Dedicated engine
Retirement accumulation + drawdown	Rare / simplified	Monthly-compounding
Old vs. new taxregime optimization	Not addressed	Dedicated Tax Engine

Implemented vs. proposed disclosed	Rarely disclosed	Explicit Phase 1/2 split
------------------------------------	------------------	--------------------------

Table I. Comparative positioning of SmartLife AI against the systems reviewed in Section II.

III. SYSTEM OVERVIEW AND ARCHITECTURE

SmartLife AI follows a conventional three-tier web architecture. The frontend is built with Next.js and React, using Recharts for interactive visualization and Axios for API communication. The backend is a FastAPI service exposing a REST API secured with JWT-based authentication (python-jose, passlib/bcrypt), with SQLAlchemy as the ORM over a relational database (SQLite for development, PostgreSQL — via psycopg — for deployment). Bank-statement ingestion uses pdfplumber for PDF parsing and a CSV/Excel path with header-alias normalization; PDF report export uses ReportLab. The data model (Table II) captures users, profiles, income, expenses, assets, liabilities, goals, investments, retirement plans, bank statements and their parsed transactions, budget plans, merchant-category overrides, a health-score history, and chat history — the last two tables reserved for the Phase 2 financial-health-score and conversational-advisory features.

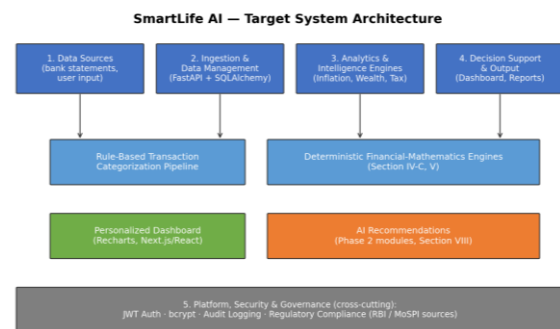


Fig. 1. SmartLife AI target system architecture

IV. IMPLEMENTED FUNCTIONAL MODULES (PHASE 1)

A. Authentication, Records, and Statement Ingestion

Users register and authenticate via JWT-secured endpoints. Generic CRUD endpoints (/records/{resource}) manage income, expense, asset, liability, goal, and investment records, validated with Pydantic models that enforce nonnegative amounts and consistent record typing. Bank statements are uploaded as PDF or CSV; the parser normalizes bank-specific column headers (date, description, amount, debit/credit, balance) via an alias dictionary and multiple date-format matchers, so statements from different banks are ingested into a single canonical transaction schema.

B. Rule-Based Transaction Categorization

Each parsed transaction description is matched against a regular-expression rule table (e.g., fooddelivery merchants map to “Food”, ride-hailing merchants map to “Transport”, brokerage/SIP keywords map to “Investments”) after normalization that strips payment-channel tokens (UPI/NEFT/IMPS/POS) and non-alphanumeric noise. Self-transfers between a user's own accounts are detected and excluded from expense totals via a separate pattern set. The categorizer is explicitly designed with an injectable fallback interface so that a learned or LLM-based classifier can be substituted in Phase 2 without changing the calling code.

C. Deterministic Financial-Mathematics Engines

Four engines implement the core inflation-aware analytics:

- Inflation Engine — $\text{future_value}(v, \pi, t) = v \cdot (1+\pi)^t$ and $\text{real_value}(v, \pi, t) = v / (1+\pi)^t$, used throughout the platform to project future costs and discount future or held sums to present purchasing power.
- Real Wealth Engine — computes net worth as assets + investments – liabilities and reports both the nominal figure and its inflation-adjusted (real) equivalent over a chosen horizon.
- Life Value Engine — computes annual surplus = 12 × (monthly income – monthly expenses), projects that surplus's future value over a chosen horizon, and nets out liabilities to produce a single “financial life value” figure.
- Retirement Engine — simulates monthly-compounding SIP accumulation with an annual step-up, converts a user-entered effective annual return into an equivalent monthly rate, then simulates year-end, inflation-indexed withdrawals during drawdown, and computes required corpus, projected corpus, readiness, and shortfall/surplus.

The required-corpus calculation for the drawdown phase (a growing annuity due at yearend) is:

$$RC = E1 \times [1 - ((1+\pi)/(1+rd))^n] / (rd - \pi), \text{ for } rd \neq \pi$$

where $E1$ is the first year's post-retirement expense, π is the assumed inflation rate, rd is the (conservative, debt-only) retirement-phase return, and n is the number of retirement years; the degenerate case $rd = \pi$ is handled separately as $RC = E1 \cdot n / (1+rd)$.

D. Income-Tax Optimization Engine (FY 2025/26)

The tax engine implements both the old and new Indian income-tax regimes for FY 2025-26 (AY 2026-27),

including age-band-specific slabs, Section 80C (₹1,50,000 cap) and Section 80D (₹25,000 / ₹50,000 senior-citizen cap) deductions, HRA exemption input, the Section 87A rebate under both regimes, and new-regime marginal relief above the ₹12,00,000 threshold, before applying a 4% health-and-education cess. A comparison function returns the better regime and the rupee savings, and an optimization function computes unused 80C/80D headroom and the marginal tax saving from using it fully — all without recommending a specific investment product.

E. Calculators and Investment Allocation

Standalone EMI, SIP, fixed-deposit, inflation-projection, and emergency-fund-target calculators are exposed as stateless POST endpoints for quick what-if analysis outside a saved financial plan. A rule-based investment-allocation recommender compares a user's current asset-class mix (equity/debt/gold) against risk-profile-specific educational targets (e.g., a “moderate” profile targets 60% equity / 30% debt / 10% gold) and returns an explainable increase/reduce/maintain action per asset class, together with a savings-rate-based readiness note; the module is explicitly labelled in-code as an explainable baseline rather than regulated investment advice.

F. Dashboard, Reporting, and Early Conversational Endpoint

A dashboard endpoint aggregates budget, wealth, and prediction data for the frontend's Recharts-based visualizations. A PDF-report endpoint (ReportLab) exports an expense report. An /ai/advice endpoint currently returns a templated, dashboard-aware reply (e.g., flagging a savings rate below 20%) and is explicitly documented in-code as a lightweight development-stage placeholder; Section VIII-D specifies its planned replacement with a retrieval-augmented Gemini-based advisory layer.

G. Testing and Validation

The backend ships with an automated pytest suite of 24 unit tests spanning four modules: financial-calculation correctness (inflation, real wealth, life value, retirement accumulation/drawdown, budget), tax-regime computation (old vs. new regime, 80C/80D optimization), bank-statement parsing (header-alias resolution, date-format matching, amount normalization across bank formats), and PDF-report generation. We run these tests using httpx-based API client fixtures against a separate database (test.db), so the verification is independent and repeatable — and it exercises

the exact same engines that produced the worked examples in Section VI.

V. MATHEMATICAL FORMULATION OF THE DETERMINISTIC ENGINES

We define NW_t as nominal net worth, π as the assumed annual inflation rate, and t as the horizon in years, so the Real Wealth Engine computes:

$$RW_t = NW_t / (1 + \pi)^t$$

In Phase 2 (Section VIII-A), this generalizes to category-aware planning, where future expenditure across C categories becomes:

$$E_t = \sum_{c=1}^C E_{c,t} \cdot (1 + \pi_c)^t$$

The Life Value Engine takes the annual surplus $As = 12 \cdot (I - E)$ and projects it forward as:

$$\text{FinancialLifeValue} = \max(0, \text{FutureValue}(As \times \text{years}, \pi, \text{years}) - L)$$

where L is outstanding liabilities. The Retirement Engine's accumulation phase compounds monthly:

$C_t = C_{t-1} \times (1 + rm) + SIP_t$, $rm = (1 + rp)^{(1/12)} - 1$ where $rp = w_E \cdot r_E + w_D \cdot r_D$ is the blended portfolio return from equity/debt weights and returns, and SIP_t steps up by a fixed percentage after each 12-month savings year. The drawdown phase applies:

$$C_t = C_{t-1} \times (1 + rd) - W_t, W_t = W_{t-1} \times (1 + \pi)$$

with readiness expressed as $\text{Projected Corpus} / \text{Required Corpus} \times 100\%$.

VI. WORKED EXAMPLES (COMPUTED DIRECTLY FROM THE DEPLOYED ENGINES)

To keep every reported figure reproducible, this section reports outputs generated by directly invoking the project's own calculation modules — no figure in this section is estimated or assumed.

A. Real Wealth over a 20-Year Horizon

Fig. 2 plots the Real Wealth Engine's output for ₹10,00,000 held at 6% p.a. inflation, showing nominal value against inflation-adjusted purchasing power.

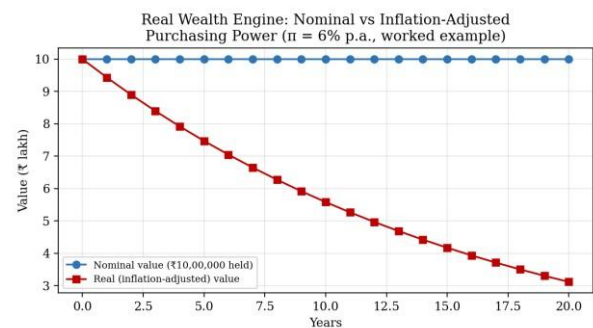


Fig. 2. Real Wealth Engine: nominal vs. inflation-adjusted value of ₹10,00,000 over 20 years at $\pi = 6\%$ p.a. — real value falls to ₹5,58,395 by year 10 and ₹3,11,805 by year 20 (generated by `real_value()`).

B. Retirement Accumulation and Drawdown

Fig. 3 and Table IV show the Retirement

Engine's output for a 28-year-old retiring at 60 with a ₹15,000 monthly SIP (10% step-up), a ₹5,00,000 existing corpus, and ₹6,00,000 current annual expenses at 6% inflation.

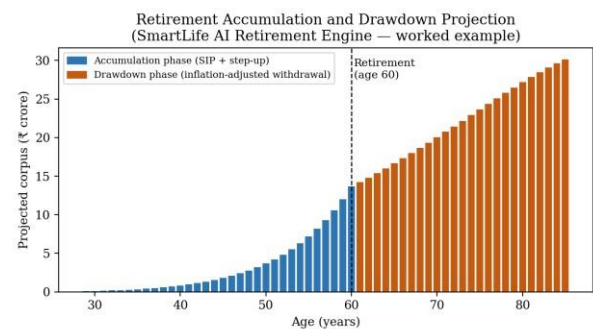


Fig. 3. Retirement Engine worked example: SIP accumulation (blue) to age 60, followed by inflationindexed drawdown (orange) to age 85 (generated by `project_retirement()`).

Quantity	Value
Projected corpus at retirement (age 60)	₹13,65,30,743
Required corpus (present value at retirement)	₹8,10,13,112
Readiness	168.5% (surplus)
Blended accumulation-phase return	10.5% p.a.

Retirement-phase return	(drawdown)	7.0% p.a.
----------------------------	------------	-----------

Table IV. Retirement Engine worked-example outputs for the scenario in Section VI-B.

C. Old vs. New Income-Tax Regime Comparison

Fig. 4 compares old- and new-regime tax liability across five income levels for FY 2025-26, with ₹90,000 (§80C) and ₹18,000 (§80D) claimed.

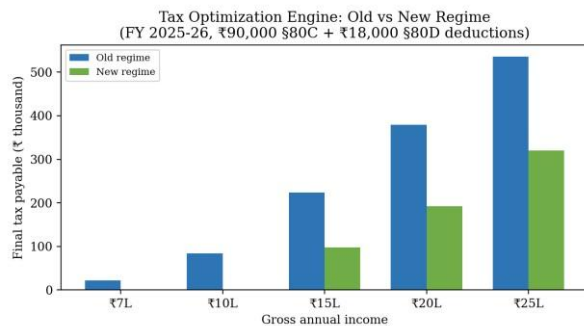


Fig. 4. Tax Optimization Engine: old- vs. new-regime final tax payable across income levels — at ₹15L income, the new regime saves ₹1,26,204 (generated by `compare_regimes()`).

VII. EXPENSE FORECASTING BASELINE AND INVESTMENT RECOMMENDER

The current expense forecaster is intentionally simple: an ordinary-least-squares linear-regression baseline (`linear_regression_forecast`) fits a trend line to a user's own historical monthly expense values, extrapolates one step ahead, and classifies the trend as increasing, decreasing, or stable relative to a noise threshold proportional to the mean. With fewer than three data points the model reports insufficient-data confidence rather than a numerically unstable forecast; with four or more points it reports medium confidence. This baseline was chosen deliberately over a higher-capacity model because most users initially have short transaction histories (a handful of months), and — consistent with the finding in the literature that model complexity should match data availability [1], [7] — a low-parameter linear model is less prone to overfitting a short series than an LSTM or a wide neural network. The investment-allocation recommender similarly uses an explainable, rulebased comparison against risk-profile targets rather than a black-box model, so that every recommendation ships with a human-readable explanation string.

VIII. PHASE 2 ROADMAP: PLANNED PREDICTIVE AND RECOMMENDATION MODULES

This section specifies, at the same level of technical precision as Sections IV–VII, the components that are designed but not yet implemented, so that the remaining half of the project has a concrete specification rather than an open-ended aspiration. Fig. 5 summarizes the implemented-vs-proposed boundary across the whole system.

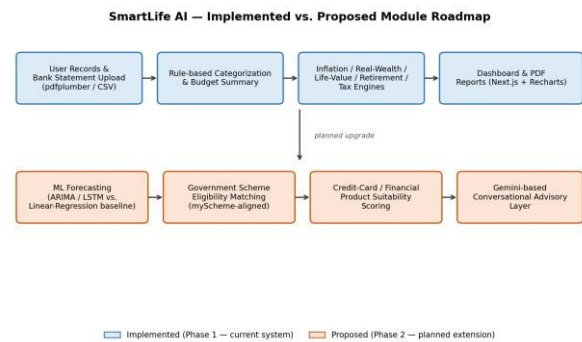


Fig. 5. Implemented (Phase 1, blue) vs. proposed (Phase 2, orange) module roadmap for SmartLife AI.

A. Comparative ML Forecasting Ensemble

Once users accumulate enough transaction history — either individually or pooled — we plan to benchmark the linear-regression baseline from Section VII against two alternatives: an $ARIMA(p,d,q)$ model, which handles autoregressive and differenced temporal patterns, and an LSTM network for capturing longer non-linear dependencies. We'll evaluate all three using a chronological holdout so no future data leaks into training, reporting MAE, RMSE, and MAPE. This also lets us directly test a claim from the literature [1], [7]: that model complexity should track dataset size, rather than always reaching for the most sophisticated architecture available.

B. Government Scheme Eligibility Recommender

For a given user (described by age, income, occupation, state, education, and employment status), each candidate scheme's mandatory conditions get checked as a binary filter, $Match_j \in \{0,1\}$. Schemes that pass are then ranked using a weighted relevance score, $S_j = \sum k w_k \cdot x_{jk}$, based on the matched criteria. We're careful not to overstate certainty here — results will be labeled "potentially eligible, verify on the official myScheme portal" [11] rather than presented as guaranteed eligibility, and we'll version the scheme database so eligibility-rule changes over time are tracked.

C. Credit-Card / Financial-Product Suitability Scoring

A suitability score $CCS_j = w_1 \cdot CB_j + w_2 \cdot AF_j + w_3 \cdot CM_j + w_4 \cdot RW_j + w_5 \cdot EL_j + w_6 \cdot OB_j$ (cashback, annual-fee fit, category-spend match, reward value, eligibility, other benefits; $\sum w_i = 1$) will rank products by fit to a user's own spending profile rather than by issuer promotion, consistent with the disclosure principles in the RBI's card-issuance framework [13]. Design patterns for combining explicit feature scoring with learned interaction terms will draw on Neural Collaborative Filtering [16] and large-scale recommender practice [15].

D. Gemini-Based Conversational Advisory Layer

The current templated /ai/advice endpoint will be replaced with a retrieval-augmented pipeline: the user's dashboard snapshot (budget, wealth, retirement readiness) will be assembled into a structured context block, combined with a system instruction constraining the assistant to the user's own data and to non-prescriptive, educational language, and submitted to the Gemini API [17]. This mirrors the system-instruction-plus-retrieved-context prompting pattern used in a related institution project for a personalized educational conversational agent [20], and will use sentence-embedding techniques such as those underlying BERT/Sentence-BERT [18], [19] if free-text financial-document Q&A is added.

IX. DISCUSSION

The central design principle demonstrated by the implemented system is that not every component of a financial-intelligence platform needs to be a machine-learned model. Inflation projection, real-wealth discounting, life-value accumulation, retirement simulation, and income-tax computation are all governed by well-defined, auditable financial mathematics and current tax law; encoding them as deterministic functions rather than learned models makes their outputs exactly reproducible (as demonstrated in Section VI, and independently verified by the automated test suite in Section IV-G) and easy to explain to a user or a viva panel. Machine learning is reserved for genuinely predictive problems — near-term expense trend extrapolation today, and inflation/CPI forecasting in Phase 2 — where no closed-form relationship exists.

This split also makes clear what's left to build. Phase 1 already handles questions like "what is my real wealth, my retirement readiness, my tax liability" using the user's own data and assumptions. Phase 2, by contrast, is scoped around questions that genuinely need pattern-learning or outside data — things like "what will I likely spend next month," "which

government schemes might apply to me," and "which financial products suit my spending." We chose to frame it this way on purpose: each Phase 2 module in Section VIII comes with a defined input, output, and evaluation method, so the remaining work is something we can actually be held to, not just a vague promise of "more AI."

X. LIMITATIONS AND FUTURE WORK

The expense forecaster is a single-feature linear baseline, not yet benchmarked against ARIMA/LSTM, as most accounts lack sufficiently long transaction history — the first Phase 2 milestone. Transaction categorization is regex-based and will misclassify merchants outside the current rule table until the documented ML/LLM extension point is used. The government-scheme, credit-card, and conversational-advisory modules are specified in Section VIII but not yet implemented, and the retirement/tax engines currently use a single deterministic assumption set rather than Monte-Carlo sensitivity bands. Automated bank-account integration is not implemented; statement upload remains manual.

XI. CONCLUSION

This paper presented SmartLife AI as it exists today: a working, deployed personal-finance platform whose deterministic inflation, real-wealth, life-value, retirement, and tax-optimization engines were exercised directly to produce every numerical result and figure in this paper, and whose correctness is independently checked by a 24-test automated suite. Rather than padding this paper with unverified machine-learning accuracy numbers, we chose to draw a clear line between what's actually built and what's planned. The Phase 2 roadmap — ARIMA/LSTM forecasting compared against our baseline, a myScheme-aligned scheme recommender, a credit-card suitability scorer, and a Gemini-based advisory layer — is laid out with specific inputs, outputs, and evaluation criteria for each piece. Our goal with this framing was to keep the project honest about where it stands right now, while still giving a concrete plan for the work that's left.

XII. ACKNOWLEDGMENT

We're grateful to our project guide, Prof. Sheetal Shimpikar, for her guidance and feedback throughout this project, and to MES's Pillai College of Engineering, New Panvel, for providing the resources to carry it out.

REFERENCES

- [1] H. Wasserbacher and M. Spindler, "Machine learning for financial forecasting, planning and analysis: Recent developments and pitfalls," *Digital Finance*, vol. 4, no. 1, pp. 63–88, 2022.
- [2] T. Bareith, T. Tatay, and L. Vancsura, "Navigating inflation challenges: AI-based portfolio management insights," *Risks*, vol. 12, no. 3, p. 46, 2024.
- [3] A. Malhotra and M. Maloo, "Understanding food inflation in India: A machine learning approach," *arXiv:1701.08789*, 2017.
- [4] B. Mahto et al., "Smart Pocket: A machine learning based expense tracker and spending predictor," *Int. J. Research and Innovation in Applied Science*, vol. 10, no. 11, pp. 352–362, 2025.
- [5] H. G. M. Harshitha, P. Kumar, and M. Rakshitha, "AI-driven financial insights for personal budget planning: A smart approach to future expense prediction," in *Proc. 3rd Int. Conf. Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, 2025.
- [6] D. Jadhav, R. Huchgond, A. Jadhav, A. Bhosale, and D. Jadhav, "AI-driven personal finance management system using machine learning and Flask," *Int. J. Scientific Research in Engineering and Management*, vol. 9, no. 11, 2025.
- [7] C. Zhang, N. N. A. Sjarif, and R. Ibrahim, "Deep learning models for price forecasting of financial time series: A review of recent advancements: 2020–2022," *WIREs Data Mining and Knowledge Discovery*, 2023.
- [8] I. Fisher, *The Money Illusion*. New York, NY, USA: Adelphi Company, 1928.
- [9] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," *Econometrica*, vol. 47, no. 2, pp. 263–291, 1979.
- [10] R. H. Thaler and C. R. Sunstein, *Nudge: Improving Decisions About Health, Wealth, and Happiness*. New Haven, CT, USA: Yale Univ. Press, 2008.
- [11] Government of India, "myScheme — National Platform for Government Scheme Discovery and Eligibility," [Online]. Available: <https://www.myscheme.gov.in/>
- [12] Ministry of Statistics and Programme Implementation (MoSPI), Government of India, "Consumer Price Index," [Online]. Available: <https://www.mospi.gov.in/>
- [13] Reserve Bank of India, "Master Direction — Credit Card and Debit Card — Issuance and Conduct Directions," Reserve Bank of India, Mumbai, India.
- [14] C. Limawan, K. W. Wahyuditomo, A. Karina, D. Fitriana, E. Irwansyah, and G. P. Kusuma, "Big data for financial analysis: Inflation rate forecasting using random forest," *Binus University*, 2022.
- [15] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. 10th ACM Conf. Recommender Systems (RecSys '16)*, 2016.
- [16] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," *Information Retrieval*, 2017.
- [17] M. Imran and N. Almusharraf, "Google Gemini as a next generation AI educational tool: A review of emerging educational technology," *Smart Learning Environments*, vol. 11, p. 22, 2024.
- [18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. 2019 Conf. North American Chapter of the Assoc. for Computational Linguistics (NAACL-HLT)*, 2019, pp. 4171–4186.
- [19] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT networks," in *Proc. 2019 Conf. Empirical Methods in Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.
- [20] M. M. Patil, Y. J. Barke, A. J. Patil, K. Rana, and R. B. Sangle, "AI-powered personalized conversational agent for education," *Int. J. Engineering Research & Technology (IJERT)*, vol. 15, no. 4, 2026.
- [21] Income Tax Department, Government of India, "Income tax slabs for FY 2025-26 (AY 2026-27) — Old and new regime," [Online]. Available: <https://www.incometax.gov.in/>
- [22] Reserve Bank of India, "Account Aggregator Framework," Reserve Bank of India, Mumbai, India, [Online]. Available: <https://www.rbi.org.in/>