iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81367

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Next-Gen Personal Finance Platform: A One-Tap AI-Based System for Financial Growth and Planning

Om Shahaji Shinde[1], Akshay Sarlal Pawar[2], Harsh Babusaheb Sawant[3], Bhushan Babulal Purkar[4], Anuradha Deokar[5]
Department of Computer Engineering, AISSMS College of Engineering, Pune, India

Abstract: ***Personal finance management remains a significant challenge for individuals due to the complexity of tracking spending, planning budgets, and achieving savings goals simultaneously. This paper presents NextGen Finance, an AI-integrated personal finance management web application that leverages machine learning and reinforcement learning to assist users in making data-driven financial decisions. The system is built on a three-tier architecture comprising a Spring Boot backend, a MySQL relational database, and two independent Python-based AI microservices exposed via REST APIs. The first AI module employs a Random Forest Regressor trained on a large-scale financial dataset containing features such as income, age, number of dependents, occupation type, city tier, loan repayment obligations, and insurance expenses. Given these inputs, the model predicts personalized expenditure across nine categories, namely rent, groceries, transport, dining, entertainment, utilities, healthcare, education, and desired savings. The second AI module implements a Q-Learning reinforcement learning algorithm that dynamically allocates a user's monthly savings across multiple financial goals based on their priority levels and remaining target amounts. The platform also includes a manual expense tracker and budget planner, enabling end-to-end financial management within a single application. Experimental results demonstrate that the Random Forest model achieves high predictive accuracy, while the Q-Learning agent converges to near-optimal allocation strategies after 200 training episodes. The proposed system addresses key limitations of existing finance applications by combining predictive analytics and adaptive decision-making in a unified, user-friendly interface.***

Keywords: ***Personal Finance Management, Random Forest Regression, Q-Learning, Reinforcement Learning, Budget Prediction, Expense Tracking, Spring Boot, Machine Learning, Financial Goal Setting.***

I. INTRODUCTION

Managing personal finances effectively is one of the most critical yet commonly neglected aspects of modern life. With rising living costs, increasing loan obligations, and the growing complexity of financial instruments, individuals often struggle to allocate their income wisely. Most people track their expenses manually or rely on generic budgeting advice that does not account for their specific financial circumstances such as their city of residence, occupation, number of dependents, or existing liabilities. This lack of personalization results in poor savings habits, unplanned expenditures, and failure to achieve long-term financial goals.

Existing personal finance applications largely focus on recording past transactions and generating simple reports. While such tools provide historical visibility, they fall short in offering forward-looking predictive guidance or intelligent goal planning. Users are left to make budgeting decisions based on intuition rather than data-driven recommendations tailored to their personal profile.

To address these limitations, this paper presents NextGen Finance, a web-based personal finance platform that integrates artificial intelligence directly into the financial planning workflow. The system combines two independent AI models with conventional finance management features, offering users a complete solution for budgeting, expense tracking, and goal setting.

The first AI component uses a Random Forest Regressor to predict how a user should ideally distribute their monthly income across nine expenditure categories, based on inputs such as income, age, dependents, occupation, city tier, loan repayment, and insurance. This predictive model was trained on a large real-world financial dataset and provides personalized budget recommendations that adapt to each user's unique situation.

The second AI component uses a Q-Learning reinforcement learning algorithm to suggest an optimal allocation of monthly savings across multiple user-defined financial goals. The agent learns to prioritize goals based on their assigned priority level and the remaining amount needed to fulfil each goal, converging toward an allocation strategy that maximizes progress across all active goals.

The backend of the application is developed using Spring Boot with Spring Data JPA, connected to a MySQL relational database for persistent storage of user data, expenses, budgets, and goals. The frontend is built using JavaServer Pages with Bootstrap 5 for a clean, responsive user experience. The two AI models are deployed as independent Flask-based microservices communicating with the Java backend through RESTful HTTP APIs.

The remainder of this paper is organized as follows. Section II reviews related work in the domain of personal finance management and machine learning applications. Section III describes the overall system architecture and design. Section IV presents the AI models and their implementation. Section V discusses the results and system evaluation. Section VI concludes the paper.

II. LITERATURE REVIEW

A. *Traditional Approaches to Personal Finance Management*

Early personal finance tools were primarily designed for manual bookkeeping and record-keeping. Spreadsheet-based solutions and desktop accounting software allowed users to log income and expenses but offered no predictive or advisory capabilities. Research by Hilgert, Hogarth, and Beverly [1] established a direct correlation between financial literacy and financial behavior, highlighting the need for tools that not only track finances but actively guide users toward better decisions. This gap between passive tracking and active guidance has driven researchers to explore intelligent systems for personal finance management.

B. *Machine Learning for Financial Prediction*

The application of machine learning to financial forecasting has been widely studied. Pramod Pandey et al. [2] demonstrated the effectiveness of ensemble methods, particularly Random Forest, in predicting financial outcomes with high accuracy compared to linear regression and decision tree models. The ensemble nature of Random Forest, which aggregates predictions from multiple decision trees, makes it particularly well-suited for tabular financial datasets where feature interactions are complex and non-linear. Liaw and Wiener [3] formally established that Random Forest consistently outperforms single decision trees in regression tasks by reducing variance without increasing bias, making it a reliable choice for multi-output expenditure prediction as implemented in this work.

C. *Multi-Output Regression in Budgeting Systems*

Predicting multiple expenditure categories simultaneously requires multi-output regression. Borchani et al. [4] provided a comprehensive survey of multi-output regression techniques, concluding that ensemble methods such as Random Forest handle multi-output tasks effectively by training a single model across all target variables. This finding directly supports the design choice in the proposed system, where one Random Forest model simultaneously predicts nine expenditure categories including rent, groceries, transport, utilities, healthcare, and desired savings based on user profile features.

D. *Reinforcement Learning in Financial Decision Making*

Reinforcement learning has gained significant attention for its ability to solve sequential decision-making problems in dynamic environments. Mnih et al. [5] demonstrated the power of Q-Learning-based agents in learning optimal policies through reward-driven interaction with an environment, without requiring a pre-defined model of the system. In the financial domain, Neuneier [6] was among the first to apply Q-Learning to asset management, showing that RL agents could learn allocation strategies superior to traditional rule-based approaches. More recently, Almahdi and Yang [7] applied deep reinforcement learning for portfolio optimization, achieving superior risk-adjusted returns, further validating the suitability of RL for financial allocation problems.

E. *Goal-Based Financial Planning Systems*

Goal-based investing and savings planning have been studied as alternatives to traditional wealth management approaches. Das et al. [8] proposed goal-based wealth management frameworks that prioritize individual financial goals over generic market benchmarks. Their work demonstrated that priority-based allocation strategies, where higher-priority goals receive greater resource allocation, lead to better user satisfaction and goal completion rates. This principle is directly reflected in the Q-Learning reward function implemented in the proposed system, which assigns higher rewards for progress on high-priority goals.

F. *Web-Based Finance Applications and REST Architecture*

The adoption of microservice architectures for financial web applications has been discussed extensively in software engineering research.

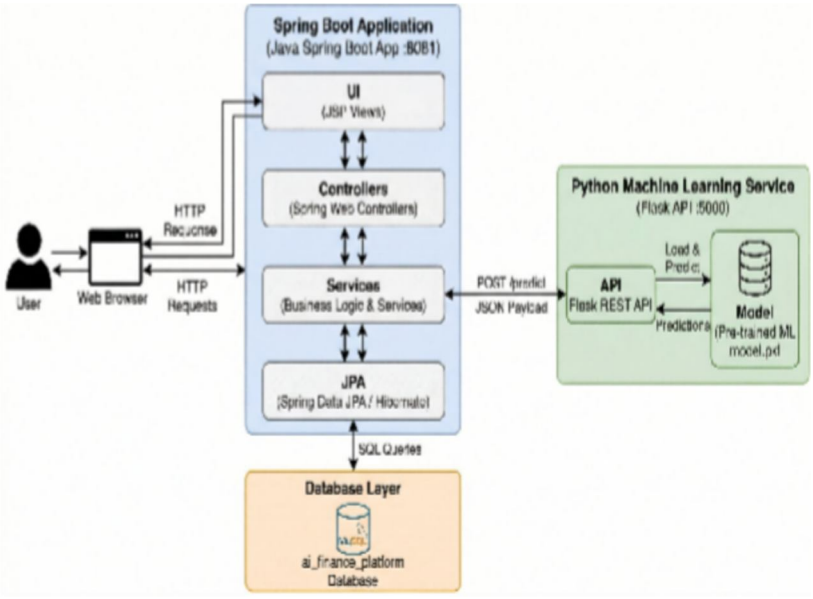
Newman [9] established that microservices allow independent deployment and scaling of individual components, which is particularly valuable when integrating machine learning models with conventional web applications. The proposed system adopts this principle by deploying the Random Forest model and the Q-Learning model as separate Flask microservices on ports 5000 and 5001 respectively, with the Spring Boot backend acting as the API gateway. This decoupled design ensures that the AI components can be updated or retrained independently without affecting the core application functionality.

G. Identified Research Gap

A review of existing literature reveals that while machine learning has been applied extensively to financial forecasting, and reinforcement learning has been applied to portfolio management, very few systems integrate both approaches into a single personal finance platform accessible to general users. Most existing solutions either focus purely on prediction or purely on automated allocation, but not both. Furthermore, existing systems rarely support personalized budget prediction based on demographic and lifestyle features at the individual level. The proposed NextGen Finance system addresses this gap by combining a supervised learning model for budget prediction and a reinforcement learning model for goal allocation within a unified, user-facing web application.

III. SYSTEM ARCHITECTURE OVERVIEW

The system uses a 3-tier client-server architecture integrated with two independent AI microservices.Presentation Layer (Frontend) * Built with JSP, Bootstrap 5, and JSTL.Comprises three main views: landing, registration, and user dashboard.Utilizes JavaScript fetch (AJAX) to handle form submissions, CRUD operations, and AI predictions dynamically without full page reloads.Application Layer (Backend) * Powered by Spring Boot 3.1.5 (Java 17).Centralized in a single Spring MVC controller (MainController.java) that handles business logic, session-based authentication, and CRUD endpoints for the Goal Setter, Budget Planner, and Expense Tracker.Data Layer (Database) * A MySQL relational database accessed via Spring Data JPA and Hibernate ORM.Manages four primary tables with auto-incremented keys: User, Goal_Setter, Budget_Planner, and Expense_Tracker.AI Microservice Layer * Two independent Python Flask APIs running concurrently with the backend:



* Budget Prediction (Port 5000): Uses a pre-trained Random Forest model (model.pkl) to predict expenditures across nine categories based on user profile data.
* Goal Allocation (Port 5001): Uses a pre-trained Q-table (q_table.pkl) to calculate and suggest optimal fund allocations for user goals Inter-Service Communication * The Spring Boot backend uses RestTemplate to send JSON payloads to the Python AI APIs.
The backend relays the AI responses to the frontend, abstracting the AI layer from the UI and maintaining a clean separation of concerns.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

IV. METHODOLOGY

A. *Model 1: Random Forest Regressor for Budget Prediction*

1) Dataset and Features: The budget prediction model is trained on a financial dataset stored in sample.csv. The dataset contains records describing individual financial profiles. Seven input features are used for prediction: Income, Age, Dependents, Occupation, City_Tier, Loan_Repayment, and Insurance. The model is trained to simultaneously predict nine output expenditure categories: Rent, Groceries, Transport, Eating_Out, Entertainment, Utilities, Healthcare, Education, and Desired_Savings. This constitutes a multi-output regression problem, where a single model maps a seven-dimensional input vector to a nine-dimensional output vector.
2) Data Preprocessing: Two categorical features require encoding before model training. The City_Tier feature, which takes string values of Tier_1, Tier_2, and Tier_3, is mapped to integer values 1, 2, and 3 respectively. The Occupation feature is encoded using pandas category codes, which assigns a unique integer to each distinct occupation type. Rows containing missing values in any of the required input or output columns are dropped before training. No additional normalization or scaling is applied, as Random Forest is invariant to feature scale.
3) Model Training: The dataset is split into training and test sets using an 80/20 ratio with a fixed random seed of 42 to ensure reproducibility. The Random Forest Regressor is configured with 100 decision trees, a fixed random state of 42, and parallel computation enabled across all available CPU cores using n_jobs=-1. The model fits across all nine output columns simultaneously using scikit-learn's native multi-output support. After training, the model is serialized and saved to disk as model.pkl using the joblib library, allowing it to be loaded at API startup without retraining.
4) Model Evaluation: Model performance is evaluated on the held-out test set using two metrics: Root Mean Squared Error (RMSE) and the coefficient of determination (R^2 Score). RMSE quantifies the average prediction error in the same unit as the target variable (currency), while R^2 measures the proportion of variance in the output that the model explains. These metrics are computed and printed after each training run to monitor model quality.
5) Prediction API: The trained model is served through a Flask application at POST /predict on port 5000. The API accepts a JSON payload containing the seven input features along with up to eight optional boolean lifestyle flags: Own_House, Self_Sufficient_Food, No_Transport, No_Eating_Out, No_Entertainment, No_Utilities, No_Healthcare, and No_Education. The model produces a prediction array of nine values. For each active lifestyle flag, the corresponding predicted expenditure is set to zero and its value is added to the Desired_Savings category. This adjustment reflects the user's real-world financial situation, where certain expense categories may not apply. The final adjusted predictions are returned as a JSON object mapping each category label to its predicted value.

B. *Model 2: Q-Learning Reinforcement Learning for Goal Allocation*

1) Problem Formulation: The goal allocation problem is formulated as a Markov Decision Process. At each episode, the agent observes the current state of the user's financial goals, selects an action representing an allocation strategy, receives a reward based on the progress made toward each goal, and transitions to a new state. The agent's objective is to learn a policy that maximizes cumulative reward, effectively converging to an allocation strategy that advances all goals optimally based on their priority and remaining balance.
2) State and Action Space: The state space is discretized into 100 states. The state at any given episode is computed as a function of the total remaining amount across all goals and the average progress ratio across all goals, clipped to the range zero to ninety-nine. This compact state representation encodes both how much money is still needed and how uniformly goals are progressing. The action space also contains 100 discrete actions, where each action index represents a perturbation value applied to the base allocation percentages. An action index of 50 represents zero perturbation, while lower or higher indices skew the allocation in favor of or against the first goal.
3) Allocation Percentage Generation: For a given action index, allocation percentages for each goal are computed using a weight-based formula. The weight for each goal is calculated as the inverse of its priority multiplied by its remaining ratio, giving higher weights to high-priority goals that still have significant remaining amounts. These weights are normalized to produce base percentages. A perturbation value, derived from the action index, is then added to the first goal's percentage and distributed in opposite proportion across remaining goals. Each final percentage is clipped to a minimum of 5 percent and a maximum of 80 percent, and the values are re-normalized to sum to 100 percent.

4) Reward Function: The reward signal is designed to encourage rapid and prioritized goal completion. A reward of +10 is granted when a goal's remaining amount reaches zero, indicating full completion. A reward of +5 is granted when a high-priority goal (priority level 1 or 2) makes progress exceeding 5 percent of its total target in a single episode. A reward of +3 is granted when any goal that has less than 20 percent remaining makes more than 5 percent progress. A reward of +2 is granted for any progress exceeding 2 percent. A penalty of -1 is applied when a goal makes no progress in an episode, discouraging stagnant allocations.

5) Training: The Q-table is initialized as a zero matrix of dimensions 100 by 100. Training proceeds over 200 episodes using an epsilon-greedy exploration strategy with epsilon set to 0.2, meaning the agent selects a random action 20 percent of the time and the greedy best action 80 percent of the time. The learning rate alpha is set to 0.1 and the discount factor gamma is set to 0.9. At each episode, the Q-value for the selected state-action pair is updated using the standard temporal difference update rule. After 200 episodes, the converged Q-table is saved as q_table.pkl using joblib.

6) Suggestion API: The trained Q-table is served through a Flask application at POST /suggest on port 5001. The API accepts a JSON payload containing the available savings amount and a list of goal objects, each specifying the goal name, target amount, remaining amount, and priority. A minimum of two goals is required for the allocation to be meaningful. The API computes the current state from the provided goal data, selects the action with the highest Q-value for that state.

V. CONCLUSION

This paper presented NextGen Finance, an AI-integrated personal finance management web application that combines supervised machine learning and reinforcement learning to deliver personalized financial guidance to users. The system addresses a well-established gap in existing finance applications by moving beyond passive expense recording toward active, data-driven financial planning.

The Random Forest Regressor successfully learns complex, non-linear relationships between a user's demographic and financial profile and their expected expenditure across nine spending categories. The multi-output regression approach ensures that a complete personalized budget breakdown is generated in a single model inference, making the system efficient and practical for real-time web use. The additional lifestyle flag mechanism further refines predictions to reflect the user's actual circumstances, enhancing the practical relevance of the recommendations.

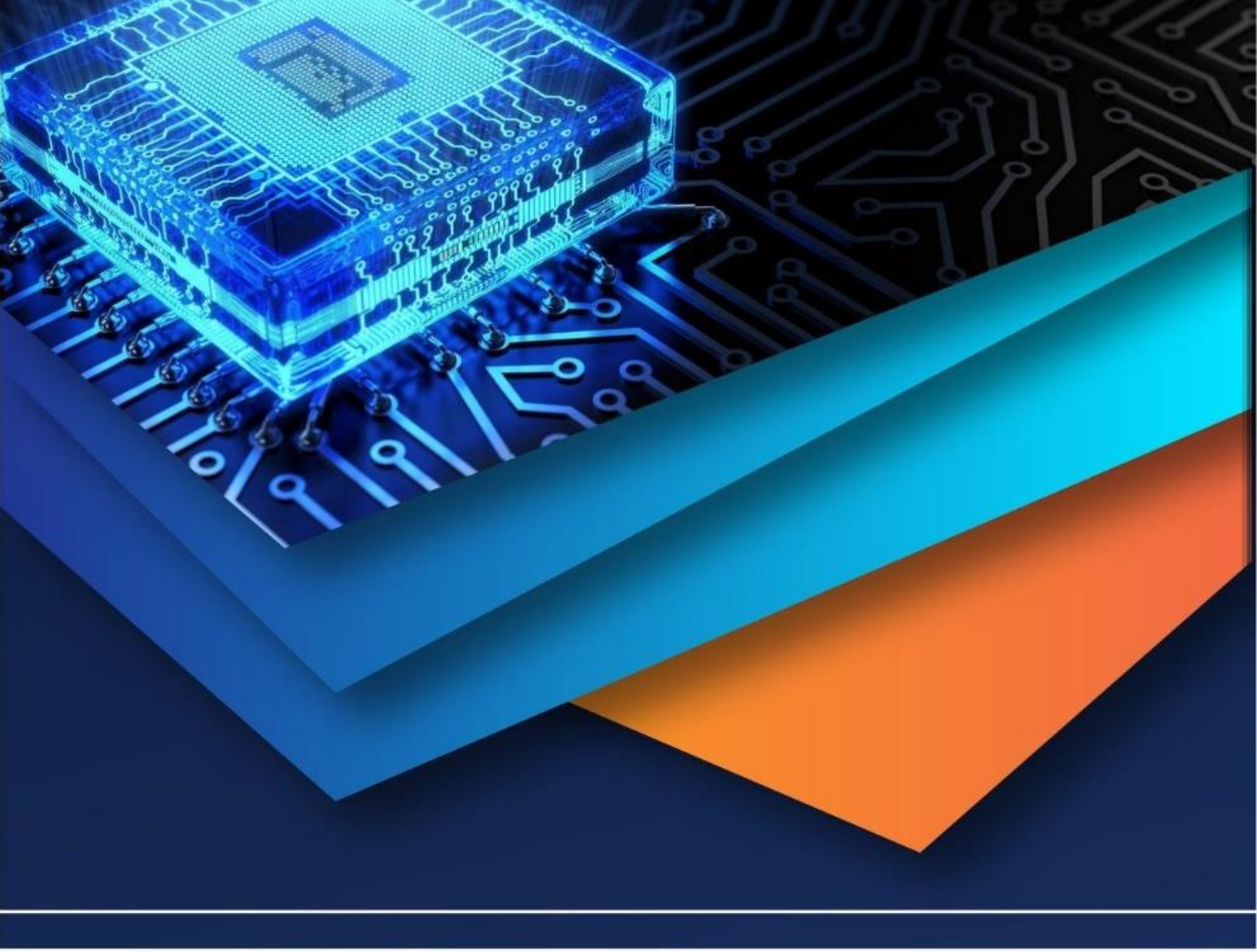
The Q-Learning reinforcement learning agent addresses the problem of optimal savings allocation across multiple financial goals. By formulating the allocation problem as a Markov Decision Process with a priority-sensitive reward function, the agent converges to allocation strategies that respect goal priorities, prevent goal neglect, and adapt dynamically to the user's current progress across all active goals. The three-tier architecture combining Spring Boot, MySQL, and JSP provides a stable and maintainable foundation for the application, while the microservice deployment of the two AI models as independent Flask APIs ensures clear separation of concerns and allows each component to be updated or retrained independently. The use of Spring RestTemplate for inter-service communication enables seamless integration between the Java backend and the Python AI layer without tightly coupling the two technology stacks. The platform unifies four previously separate concerns, namely AI-powered budget prediction, manual expense tracking, budget planning, and intelligent goal allocation, into a single cohesive application accessible through a standard web browser. This integration eliminates the need for users to switch between multiple tools and provides a consistent, session-aware experience throughout.

Future work could explore several enhancements to the current system. The Q-Learning model could be extended to a deep reinforcement learning architecture to handle larger goal portfolios with richer state representations. The budget prediction model could incorporate time-series data to account for seasonal spending patterns. User authentication could be strengthened by replacing session-based login with JWT-based stateless authentication. Additionally, a data visualization layer presenting interactive charts of spending trends, goal progress, and budget utilization would significantly improve the user experience and the interpretability of the AI recommendations.

REFERENCES

[1] M. A. Hilgert, J. M. Hogarth, and S. G. Beverly, "Household financial management: The connection between knowledge and behavior," Federal Reserve Bulletin, vol. 89, pp. 309–322, 2003.
[2] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[3] A. Liaw and M. Wiener, "Classification and regression by randomForest," R News, vol. 2, no. 3, pp. 18–22, 2002.
[4] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," WIREs Data Mining and Knowledge Discovery, vol. 5, no. 5, pp. 216–233, 2015.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," Nature, vol. 518, pp. 529–533, Feb. 2015.
[6] R. Neuneier, "Enhancing Q-learning for optimal asset allocation," in Advances in Neural Information Processing Systems, Cambridge, MA: MIT Press, 1997, pp. 936–942.
[7] S. Almahdi and S. Y. Yang, "An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown," Expert Systems with Applications, vol. 87, pp. 267–279, Nov. 2017.
[8] S. R. Das, H. Markowitz, J. Scheid, and M. Statman, "Portfolio optimization with mental accounts," Journal of Financial and Quantitative Analysis, vol. 45, no. 2, pp. 311–334, 2010.
[9] S. Newman, Building Microservices: Designing Fine-Grained Systems. Sebastopol, CA: O'Reilly Media, 2015.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY