

# NeuroVerse: An AI-Driven Peer-to-Peer Learning and Mentorship Platform

<sup>1</sup>Mrs. V. Christy, <sup>2</sup>Goutham M, <sup>3</sup>Vishwas D, <sup>4</sup>Suhas A, <sup>5</sup>K. Varun Krishna, <sup>6</sup>Masiha Sadaf

<sup>1</sup>Assistant Professor, Dept of CSE (AI & ML), East Point College of Engineering and Technology, Bengaluru

<sup>2</sup>Student, Dept of CSE (AI & ML), East Point College of Engineering and Technology, Bengaluru

<sup>3</sup>Student, Dept of CSE (AI & ML), East Point College of Engineering and Technology, Bengaluru

<sup>4</sup>Student, Dept of CSE (AI & ML), East Point College of Engineering and Technology, Bengaluru

<sup>5</sup>Student, Dept of CSE (AI & ML), East Point College of Engineering and Technology, Bengaluru

<sup>6</sup>Student, Dept of CSE (AI & ML), East Point College of Engineering and Technology, Bengaluru

\*Corresponding author email: [mgoutham212@gmail.com](mailto:mgoutham212@gmail.com)

**Abstract** — Most online learning systems deliver course material well, but they do little to connect a learner with a suitable mentor, check what the learner already knows, or give people a reason to return after the first few sessions. This paper presents NeuroVerse, a web platform for peer-to-peer learning in which every user can both teach and learn. Learners take a skill assessment, are matched to mentors by a transparent weighted score, and book paid sessions using in-platform credits, with an AI study assistant available between sessions. The platform uses Next.js, React and Tailwind CSS on the front end, Next.js API routes running on Node.js, and a PostgreSQL database hosted on NeonDB. Accounts are protected with JSON Web Tokens stored in HTTP-only cookies and bcryptjs password hashing. Assessment questions are drawn at random from a local bank and graded by exact match for multiple-choice items and by a keyword heuristic for code, and a threshold on the percentage score assigns a beginner, intermediate or advanced level. Mentors are ranked on a 100-point score built from skill, level, language, rating and availability. A Gemini 1.5 Flash model powers the study assistant, Jitsi Meet links host the sessions, and daily riddles with credit rewards encourage repeat visits. The prototype was checked manually, and no automated tests or user study have been carried out. Credit purchases are simulated, submitted code is not executed, and several production features, such as real payments, mentor availability and access control, remain to be built.

**Keywords** — AI study assistant; credit-based learning; gamification; mentor matching; peer-to-peer learning; skill assessment.

## I. INTRODUCTION

Online education has put study material within reach of almost anyone with an internet connection, but access to material is not the same as being taught. Many platforms still offer little beyond recorded lectures and quizzes. There is no mentor to ask, no dependable way to measure what a learner can do, and no shared space for working through problems together. Learners end up using one tool to find a tutor, another to take a test and a third for video calls, which scatters the experience. Peer-to-peer learning, in which participants teach as well as learn, is one response to this [1].

Three gaps stand out. First, finding a mentor whose skills fit a specific goal is difficult, because most platforms offer search filters rather than any ranking of fit [2]. Second, skill levels are usually self-reported, so neither the mentor nor the learner knows where to start. Third, a learner who is stuck while no mentor is online has nowhere to turn, since few platforms include AI-based study help [3]. Peer-to-peer designs also need a way to keep mentors active once the initial enthusiasm fades.

This paper describes NeuroVerse, a full-stack prototype that brings these pieces into a single application. Its main contributions are:

- A seven-layer architecture, from user roles to external services, built on Next.js, Node.js and PostgreSQL.

- A skill assessment that classifies users as beginner, intermediate or advanced from a percentage score and passes that level on to mentor matching.
- A rule-based, explainable mentor-matching score out of 100, together with a three-stage learning path for the chosen skill.
- A credit economy in which learners spend and mentors earn credits through priced, transaction-safe bookings, plus daily riddles that reward return visits.
- A Gemini-based study assistant, chat, file sharing and Jitsi Meet sessions inside the same application.

The remainder of the paper is organised as follows. Section II reviews related work. Section III describes the system design and methods. Section IV presents the implemented features, results and discussion. Section V concludes and lists future work.

## II. RELATED WORK

*E-learning and recommendation.* Most e-learning systems follow a familiar pattern of recorded lectures, reading material and an end-of-module quiz. That is cheap and flexible, but it leaves little room for personal interaction. Course recommendation on such platforms has typically relied on collaborative filtering, content-based filtering or hand-written rules [4]–[7]. These methods work once enough usage data has

built up, but they are tuned to what is popular and say little about what a particular learner can actually do or how that person prefers to learn.

*AI and peer learning.* Adding machine learning has partly closed this gap, because models that examine behaviour, performance and stated interests can describe a learner far better than click history alone. Peer-to-peer platforms have grown for a related reason: participants both consume and produce knowledge instead of watching a finished course [1]. Even so, pairing the right mentor with the right learner, and judging whether a session was worth it, remain open problems. Work on AI-enhanced peer assessment suggests that AI and human mentors work best together [3], and work on question sequencing shows that item difficulty is often estimated from learner behaviour instead of being fixed by hand [8].

*Gamification and infrastructure.* Two further strands are relevant. Points, streaks and credits have been shown to counter the weak motivation that undermines much self-paced learning [9], [10]. On the engineering side, educational tools increasingly use cloud-native, API-first designs in place of a single monolithic server, which makes it practical to combine real-time communication, AI inference and a relational database in one product.

NeuroVerse draws on both strands by pairing a credit-based reward loop with a Next.js and PostgreSQL back end. Unlike the learned recommenders in [4]–[6], its mentor matching is a transparent weighted-scoring rule, chosen because a new platform has no interaction history to train a model on. It also differs from course recommenders in which assessment and recommendation are handled separately [2], [5]: here the classified skill level feeds straight into mentor ranking.

### III. METHODOLOGY

NeuroVerse was built as one full-stack web application. This section describes its architecture and the methods used for authentication, assessment, mentor matching, pricing and booking, and for the supporting study-assistant, chat and gamification features.

#### A. System Architecture

The platform is organised as seven layers, shown in Fig. 1, running from the user roles at the top to the database and third-party services underneath.

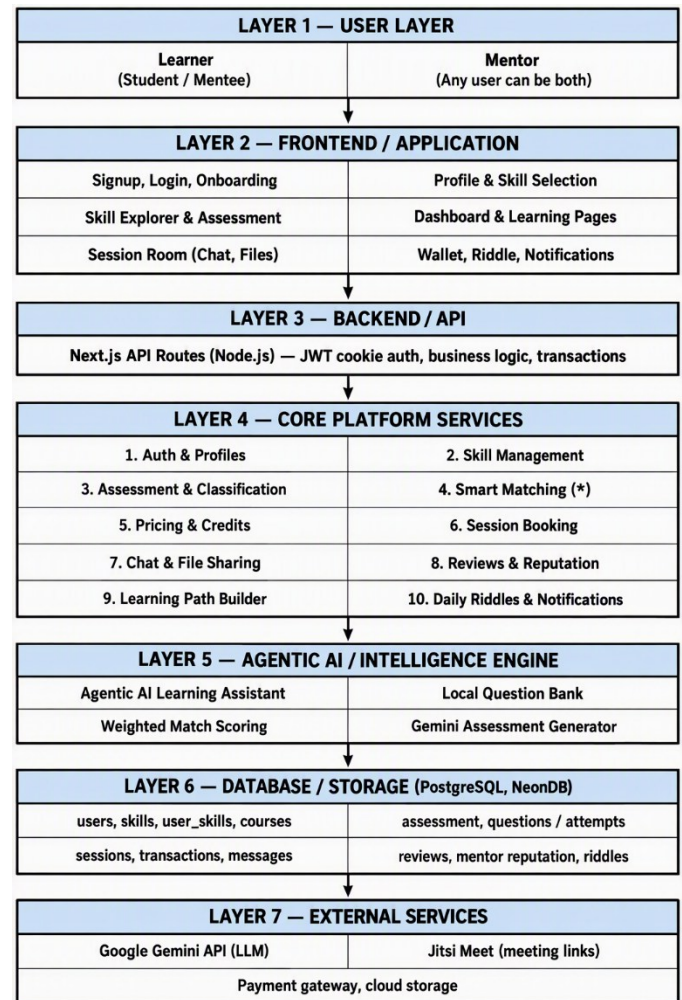


**Fig. 1.** Layered architecture of the NeuroVerse platform. (\*) marks the module at the centre of the platform. The payment gateway and cloud file storage in Layer 7 are planned, and the Gemini assessment generator in Layer 5 is implemented but not currently active.

Layer 1 separates learners from mentors, although both use one application and any user can hold either role. Layer 2 is the front end: registration, login and onboarding, profile management, skill exploration and assessment, the dashboard, the session room with chat, files and the study assistant, the wallet, the daily riddle and notifications. Every action from these screens goes through the Next.js API routes of Layer 3, which handle authentication, business logic and database transactions.

Most of the application logic sits in the ten modules of Layer 4: authentication and profiles, skill management, assessment and classification, smart matching, pricing and credits, session booking, chat and file sharing, reviews and reputation, learning-path building, and riddles and notifications. Layer 5 holds the intelligence: the Gemini-powered study assistant, the local question bank, the weighted matching score, and a Gemini assessment generator that is implemented but not currently active. Users, skills, assessment results, sessions, messages, reviews and credit transactions are stored in Layer 6 (PostgreSQL on NeonDB). Layer 7 contains

the external services in use, the Gemini API and Jitsi Meet; a payment gateway and cloud file storage are planned.

Two design decisions shaped the architecture. Smart matching sits at the centre of Layer 4 because pairing the right learner with the right mentor is the main value the platform offers. Every credit movement is also written as a row in a dedicated transactions table, and a booking updates both balances inside one database transaction with row-level locks, which keeps the teach-to-earn, spend-to-learn loop easy to audit.

### B. Technology Stack, Data Model and Authentication

Table I lists the technologies used. The front end is built with Next.js 14 and React, styled with Tailwind CSS. The back end runs on Node.js through Next.js API routes, so the whole application lives in one codebase. Data is kept in a PostgreSQL database hosted on NeonDB, which holds 15 tables: users, skills, user\_skills, assessment\_questions, assessment\_attempts, courses, mentor\_reputation, sessions, transactions, reviews, notifications, session\_files, messages, riddles and riddle\_attempts. The schema uses foreign keys, unique constraints, indexes and UUID keys generated with `gen_random_uuid()`.

TABLE I. TECHNOLOGY STACK

| Layer          | Technologies                                                                        |
|----------------|-------------------------------------------------------------------------------------|
| Frontend       | Next.js 14.2 (Pages Router), React 18, JSX, Tailwind CSS 3.4, PostCSS, Autoprefixer |
| Backend        | Next.js API routes, Node.js, formidable (file uploads), uuid                        |
| Database       | PostgreSQL on NeonDB, node-postgres (pg) with connection pooling, pgcrypto          |
| Authentication | JWT (jsonwebtoken), bcryptjs, HTTP-only cookies (cookie)                            |
| AI integration | Gemini 1.5 Flash API (study assistant; assessment generator, inactive)              |
| Meetings       | Jitsi Meet links (no embedded SDK)                                                  |
| Configuration  | dotenv, environment variables (DATABASE_URL, JWT_SECRET, GEMINI_API_KEY)            |

Every learner and mentor registers with a name, email and password. Passwords are hashed with bcryptjs at registration, and a signed JSON Web Token (JWT) with a seven-day expiry is issued in an HTTP-only cookie. Signup, login, logout and current-user routes are provided. During onboarding, users record their age, a short biography, spoken languages, the skills they can teach, the skills they want to learn and a target skill level. Each new account receives 50 credits. Email verification, password reset, OAuth, two-factor authentication and login rate limiting are not yet implemented.

A separate Python (FastAPI) notebook connects to the same database and reproduces the classification, pricing and matching logic for testing and demonstration. It is not part of the production runtime.

### C. Skill Assessment and Classification

Users take an assessment to establish their level in a skill. Questions come at three difficulty levels, beginner,

intermediate and advanced, worth 10, 15 and 20 points respectively. Multiple-choice and code questions are chosen at random from a local question bank covering Python, JavaScript, SQL and English, with generic fallback questions for other skills, using Node's cryptographic random-integer function.

Multiple-choice answers are graded by a trimmed, case-insensitive exact match. Code answers are graded by a lightweight heuristic: the submission is lowercased, comments are stripped, whitespace is normalised, and the result is checked for required keywords, key operators and selected aliases (for example, `fib` and `fibonacci`). The code is not executed. The percentage score  $p$  sets the level: advanced for  $p \geq 80$ , intermediate for  $50 \leq p < 80$ , beginner for  $20 \leq p < 50$ , and not assessed for  $p < 20$ . Results are stored in the `assessment_attempts` and `user_skills` tables. A Gemini-based generator that requests six validated questions (two per level, four-option multiple choice) has also been implemented, but the active assessment route uses the local bank.

### D. Mentor Matching and Learning Path

Mentors are ranked for a learner with a weighted suitability score out of 100, using the weights in Table II. The score takes into account the requested skill, the learner's target level, the mentor's level, the languages of both people, the mentor's average rating and preferred availability, and it returns a short, human-readable explanation of each match. Matching is a rule-based method, not a trained model. Because the database has no mentor availability field yet, the availability component currently awards neutral points. For the chosen skill the system also builds a three-stage learning path: fundamentals, applied practice and advanced mastery.

TABLE II. MENTOR-MATCHING SCORE WEIGHTS

| Matching factor   | Weight |
|-------------------|--------|
| Exact skill match | 40%    |
| Mentor level fit  | 20%    |
| Language overlap  | 15%    |
| Mentor rating     | 15%    |
| Availability      | 10%    |

### E. Session Pricing and Booking

Learners book mentoring sessions by spending credits [1]. A booking records the course, mentor, learner, number of hours, scheduled start time, price breakdown, status and a meeting link. The price per hour is

$$P = R \times f_{\text{level}} \times f_{\text{skill}} \times f_{\text{mentor}} \quad (1)$$

where  $R$  is the base rate of 20 credits per hour and the three factors are given in Table III. The learner pays the base cost plus a 10% platform fee, and the mentor receives the base cost minus a 10% deduction, with all values rounded to two decimals:

$$C_{\text{learner}} = 1.10 \times P \times h, \quad C_{\text{mentor}} = 0.90 \times P \times h \quad (2)$$

where  $h$  is the number of hours booked.

TABLE III. SESSION PRICING FACTORS

| Factor                               | Value                                                                                            |
|--------------------------------------|--------------------------------------------------------------------------------------------------|
| Base rate $R$                        | 20 credits/hour                                                                                  |
| Learner level $f_{\text{level}}$     | Beginner 0.5, Intermediate 1.0, Advanced 1.5                                                     |
| Skill difficulty $f_{\text{skill}}$  | Per-skill multiplier (e.g., 1.0–1.4)                                                             |
| Mentor expertise $f_{\text{mentor}}$ | Rating $\geq 4.5$ : 1.5; 3.5 to $<4.5$ : 1.2; 2.0 to $<3.5$ : 1.0; $<2.0$ : 0.5; no reviews: 1.0 |

A booking runs as a single PostgreSQL transaction. It locks the course row, calculates the price, locks the learner's balance, checks that the credits are sufficient, deducts them from the learner, credits the mentor, creates the session, records a transaction for each party, creates notifications and commits. Meetings use unique Jitsi Meet URLs generated with a UUID and opened in the browser. The current version does not check for conflicting bookings, past dates, maximum length, mentor working hours or time zones, and it does not support rescheduling, cancellation or refunds.

#### F. Study Assistant, Collaboration and Gamification

The Study Buddy calls the Google Gemini 1.5 Flash API through a server-side proxy, so the API key stays on the server, to answer questions and give general study help to learners and mentors. Each request carries the course title as context along with the learner's question. The assistant does not yet keep conversation history, and access is not limited to session members.

Inside each session page, mentors and learners exchange text messages that are stored in PostgreSQL and fetched through API requests; there is no WebSocket or push channel yet. Users can upload files of up to about 100 MB, which are kept on the server's local filesystem with metadata in the database. In-app notifications are created for bookings, confirmations and reviews. After a session, the learner can submit a 1–5 rating with written feedback, which updates the mentor's average rating, review count and expertise multiplier and therefore affects future prices.

A daily riddle widget picks a riddle related to the user's known skills, falling back to generic riddles, and skips riddles the user has already answered correctly. Each correct answer earns 1 credit, up to 5 credits per calendar day, and every attempt is logged. The dashboard shows the user's skill mix and credit balance, and mentors earn credits by teaching.

#### G. System Workflow

A user typically moves through the platform in this order: registration, skill selection, assessment, skill classification, mentor matching, session booking [1], AI-assisted learning [3] and review and feedback [3]. Assessment and classification come before matching so that matching can rely on a measured skill level rather than a self-reported one. The assessment is recommended but optional: a learner may go straight to mentor matching, in which case the level declared during onboarding is used. The study assistant and the review step remain available around the session that follows.

## IV. RESULTS AND DISCUSSION

### A. Implementation Status

NeuroVerse was developed and tested as a web-based prototype. It brings authentication, AI-assisted learning, mentor matching, session booking, assessment tracking and gamification into one environment. Table IV shows which features are fully implemented, which are only partly implemented, and which are simulated or absent in the current release.

TABLE IV. IMPLEMENTATION STATUS

| Status              | Features                                                                                                                                                                                                                                                                                                                       |
|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Implemented         | Authentication, profiles and onboarding, local assessment generation, multiple-choice and heuristic code grading, level classification, weighted mentor matching, learning path, credit pricing and booking transaction, notifications, reviews, daily riddles, session chat, file upload, Gemini study assistant, Jitsi links |
| Partial             | Gemini assessment generator (not the active path), mentor availability, scheduling and review validation, session access control, real-time updates                                                                                                                                                                            |
| Simulated or absent | Credit purchase, code execution, embedded video, cloud storage, WebSocket messaging, automated tests, translation, sentiment analysis, learned recommendation                                                                                                                                                                  |

Core learner workflows were checked manually across the dashboard, recommendation and skill-matching modules. No automated test suite or formal user study has been carried out, so the results below describe functional behaviour, not measured performance.

### B. Main Screens and Computed Results

Figs. 2–7 show the main screens. The dashboard (Fig. 2) gathers the credit balance, progress, upcoming sessions and the daily riddle in one place, and skill exploration (Fig. 3) lets learners browse skills and look for mentors.

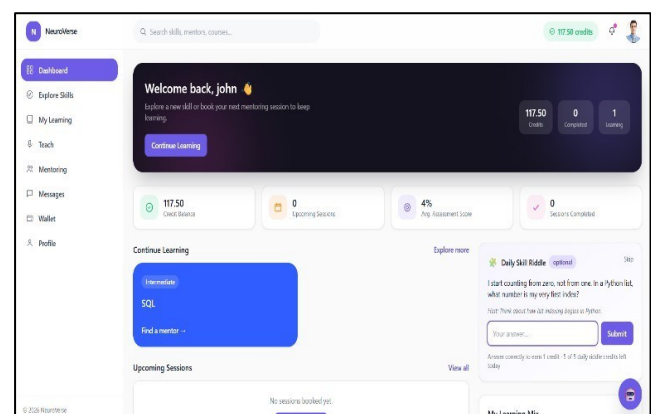


Fig. 2. Learner Dashboard

The Fig. 2. Learner Dashboard depicts a Neuroverse dashboard portraying a user's credit balance, progress in lessons, upcoming sessions, assessment score, learning

mix which indicates the percentage of the number of problems at various difficulty levels, and daily skill riddle to earn more credits.

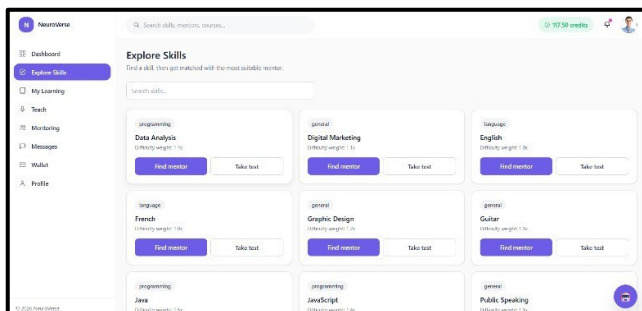


Fig. 3. Skill Exploration Interface

This Fig. 3. Skill Exploration Interface presents the skill discovery module, which facilitates students in examining the available skills, identifying appropriate mentors, and assessing skills of their learning interests.

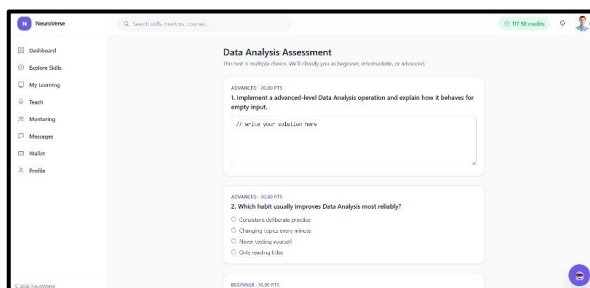


Fig. 4. The interface of skill assessment

This Fig. 4. The interface of skill assessment. The figure represents the module which identifies users' knowledge level via skill-related questions and defines their skill level in order to individualize the mentor matching process.

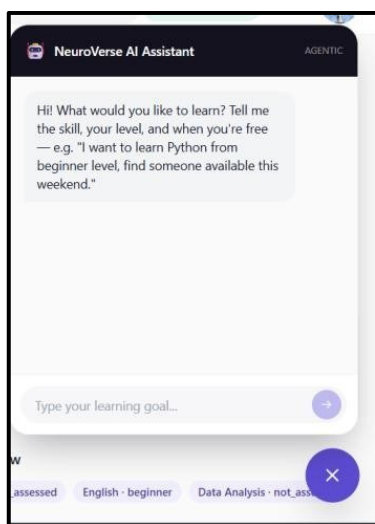


Fig. 5. AI-Powered Mentor Finding and Matching

Fig. 5. AI-Powered Mentor Finding and Matching shows the mentor discovery interface, which enables learners to discover suitable mentors according to their skill requirements and proficiency levels facilitating personalized peer-to-peer learning through intelligent skill matching.

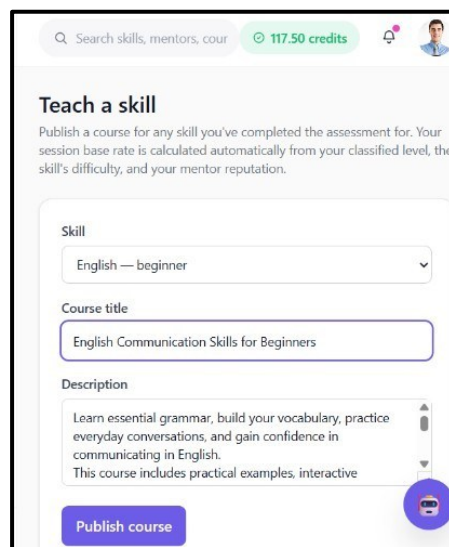


Fig. 6. Skill Teaching Interface

The Fig. 6. Skill Teaching Interface illustrates the skill publication module, through which users can create and publish learning offerings, enabling them to share their knowledge and participate in reciprocal peer-to-peer learning.

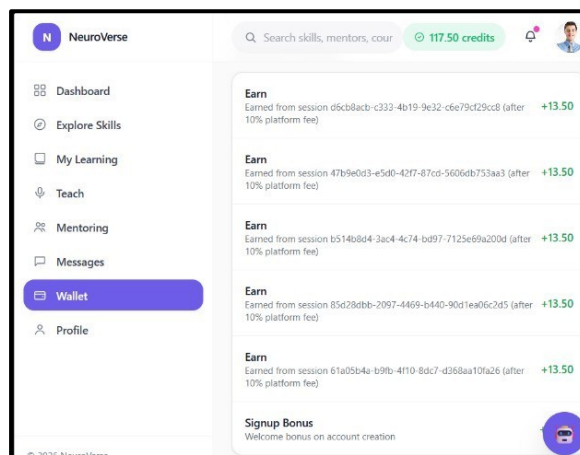


Fig. 7. Credit wallet, showing the available credits and the transaction history for learning and mentoring activity.

Fig. 8 shows the hourly learner cost computed from (1) and (2) with  $f_{\text{skill}} = 1.0$  and  $h = 1$ . The cost runs from 5.5 to 49.5 credits. Learner level and mentor rating each scale the price: at every level, a mentor rated 4.5 or above costs three times as much as one rated below 2.0. The 50-credit welcome balance just covers one hour at the highest tier.

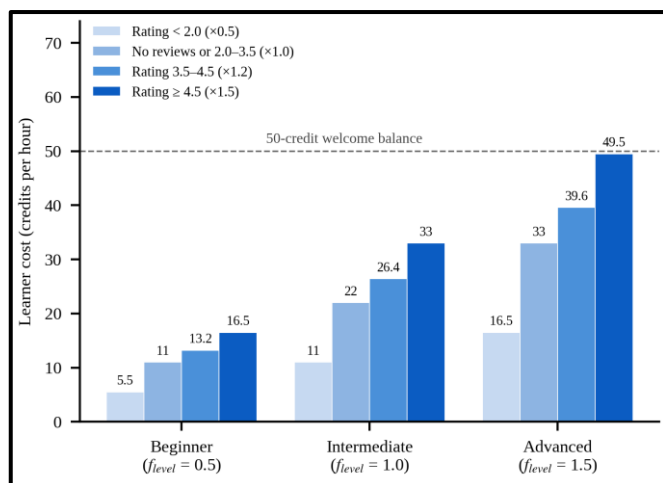


**Fig. 8.** Hourly learner cost

Fig. 8. shows Hourly learner cost (credits) by learner level and mentor-rating tier, computed from (1) and (2) with  $f_{\text{skill}} = 1.0$  and  $h = 1$ . Cost ranges from 5.5 to 49.5 credits.

Fig. 9 applies the weights in Table II to three hypothetical mentors, with availability left out so that the maximum is 90. Mentor A (exact skill, matching level, full language overlap, rating 4.8) scores 89.4, Mentor B (exact skill, adjacent level, full language overlap, rating 4.2) scores 77.6, and Mentor C (related skill, matching level, partial language overlap, rating 4.9) scores 62.2. Mentor C has the highest rating but ranks last, because the 40-point skill match dominates the score.

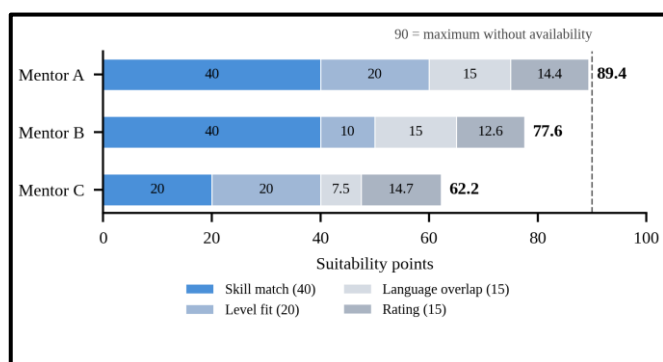


**Fig. 9.** Illustrative mentor-suitability

Fig. 9. Shows Illustrative mentor-suitability scores for three hypothetical mentors, broken down by component (availability omitted). Weights are from Table II; sub-scores are assumed illustrative values.

### C. Discussion

NeuroVerse lets every user act as both learner and mentor, in contrast to systems in which teachers guide students from the top down. This fits research on peer-to-peer platforms [1], but it makes matching harder than course recommendation, because the system has to weigh what the learner wants against what the mentor can offer. Once a learner states a target skill, level and languages, the weighted score is meant to narrow the field to a few relevant mentors, so that nobody has to sift through dozens of irrelevant results.

Building the assessment was one of the hardest parts of the project. Producing multiple-choice and coding questions at three consistent difficulty levels proved difficult. A Gemini 1.5 Flash generator was implemented with output validation (six questions, two per level, four options each), but the active flow draws from a curated local bank. This echoes question-sequencing work, where difficulty is often estimated from behaviour and not fixed by hand [8]. In NeuroVerse the classified level also feeds matching, unlike course recommenders in which assessment and recommendation are handled separately [2], [5]. The classifier cut-offs are fixed for now, and calibrating them against real assessment data is planned.

The credit system is meant to be the platform's main behavioural lever. Purely volunteer-driven peer-learning platforms often struggle when initial excitement fades and mentor participation drops. By letting users earn credits through mentoring and riddles and then spend them on sessions, NeuroVerse creates a simple incentive loop, in line with gamification studies showing that rewards can sustain engagement [9], [10]. Because credit activity is recorded in its own transactions table, unusual activity can be logged and audited without disturbing the rest of a user's data. The real effect of the credit system will only be clear once user data exists.

The AI study assistant was added alongside peer mentoring to fill the gap when a learner is stuck and no mentor is available. The assistant clarifies concepts and answers practice questions, while mentors guide the deeper learning that needs judgement and feedback, which matches the view that AI and human mentors work best together [3].

Using Next.js for both front end and back end made development easier for a small team, but it postponed some scaling decisions; the AI pipeline, for example, was not separated into its own service. JWT authentication in HTTP-only cookies was chosen for simplicity and suits the current scope, although server-side session revocation would be needed for multi-device use. The separation between front end, back end, AI components and storage is intended to keep the system maintainable, so that new modules and external services can be added at Layers 4, 5 or 7 without disturbing the layers below.

### D. Limitations

The current version has several limitations. Credit purchases are simulated and no payment provider is integrated. Code answers are matched heuristically and not run in a sandbox, so correct but unconventional solutions may be misgraded. Mentor availability is not modelled, and booking does not check for conflicts or past dates. The daily riddle credit check is not protected by a single transaction, so simultaneous requests could exceed the daily limit. Reviews are not restricted to completed sessions or to one per session. Uploaded files are stored locally and are publicly accessible, and the chat and study-assistant routes do not enforce session membership. Matching accuracy at scale and the long-term effect of the credit system can only be assessed once real user data is available.

### V. CONCLUSION AND FUTURE WORK

NeuroVerse is a prototype peer-to-peer learning and mentorship platform that combines skill assessment, weighted

mentor suggestion, credit-based session management, an AI study assistant and communication tools in one application. The prototype shows that skill-based matching, level-based assessment and incentive-based engagement can work together on a modern web stack of Next.js, PostgreSQL and the Gemini API.

Future work will focus on making the platform ready for production: a real payment gateway; sandboxed code execution for assessments; enabling the validated Gemini assessment generator; a mentor availability calendar with conflict and past-date checks, cancellation and refunds; WebSocket messaging and an embedded video interface; cloud file storage with private links; session-level access control; unique reviews tied to completed sessions; email verification, password reset and login rate limiting; and automated unit and integration tests. On the AI side, we plan sentiment analysis of feedback, learned recommendations once enough usage data exists, and a user study to evaluate matching quality and the credit system.

We thank our guide Mrs. V. Christy, Assistant Professor, Department of CSE (AI & ML), EPCET, Bengaluru, for her guidance and support throughout the design and development of this work.

## VI. REFERENCES

- [1] V. Anantha, "Peer to peer learning platform optimized with machine learning," arXiv preprint arXiv:2209.03489, Sep. 2022. [Online]. Available: <https://arxiv.org/abs/2209.03489>
- [2] R. Soni, B. Suleiman, and S. Singh, "Improving graduate outcomes by identifying skills gaps and recommending courses based on career interests," arXiv preprint arXiv:2511.09819, Nov. 2025. [Online]. Available: <https://arxiv.org/abs/2511.09819>
- [3] K. J. Topping, E. Gehringer, H. Khosravi, S. Gudipati, K. Jadhav, and S. Susarla, "Enhancing peer assessment with artificial intelligence," *Int. J. Educ. Technol. High. Educ.*, vol. 22, Art. no. 3, 2025, doi: 10.1186/s41239-024-00501-1.
- [4] D. Rodríguez, M. A. Sicilia, S. Sánchez-Alonso, L. Lezcano, and E. García-Barriocanal, "Exploring affiliation network models as a collaborative filtering mechanism in e-learning," *Interact. Learn. Environ.*, vol. 19, no. 4, pp. 317–331, 2011, doi: 10.1080/10494820903148610.
- [5] V. Anupama and M. S. Elayidom, "Course recommendation system: Collaborative filtering, machine learning and topic modelling," in *Proc. 8th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Coimbatore, India, 2022, pp. 1459–1462, doi: 10.1109/ICACCS54159.2022.9785353.
- [6] N. B. Salehudin, H. Kahtan, H. Al-bashiri, and M. A. Abdulgabbbar, "A proposed course recommender model based on collaborative filtering for course registration," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 11, pp. 162–168, 2019, doi: 10.14569/IJACSA.2019.0101122.
- [7] M. Recker, A. Walker, and D. Wiley, "Collaboratively filtering learning objects," in *Designing Instruction with Learning Objects*, D. A. Wiley, Ed. Bloomington, IN, USA: AECT, 2001, pp. 243–259.
- [8] L. Sidi and H. Klein, "Neural network-based collaborative filtering for question sequencing," arXiv preprint arXiv:2004.12212, Apr. 2020. [Online]. Available: <https://arxiv.org/abs/2004.12212>
- [9] E. M. Jianu and A. Vasilateanu, "Designing of an e-learning system using adaptivity and gamification," in *Proc. IEEE Int. Symp. Syst. Eng. (ISSE)*, 2017, pp. 1–4, doi: 10.1109/SysEng.2017.8088270.
- [10] E. Lavoué, B. Monterrat, M. C. Desmarais, and S. George, "Adaptive gamification for learning environments," *IEEE Trans. Learn. Technol.*, vol. 12, no. 1, pp. 16–28, 2019, doi: 10.1109/TLT.2018.2823710.