Ada Lovelace-Moores

Full-Stack Time Traveler

Contact

ada@futurecomputing.dev

+1-800-FIRST-BUG

originalcoder

Location

Silicon Valley via Victorian London

Work Experience

January 2020 – Today

5 years, 2 months

Principal Algorithm Architect at Temporal Paradox Solutions :

Developing computational models that predict their own development

- Designed a self-evolving neural network that somehow wrote its own documentation (a technological miracle).
- Created 'Babbage 2.0', a quantum computing framework that solved problems before they were submitted, reducing customer wait times to negative 3 days.
- Led a team of 8 engineers in developing time-synchronized databases that maintain consistency across parallel timelines.
- Pioneered 'Analytical Prediction Engine™', a service that has predicted 97% of Silicon Valley startups before their founders even thought of them.

March 2015 – December 2019

4 years, 9 months

Debugging Specialist at Retrograde Technologies :

Fixed bugs in historical computing systems that never existed

- Refactored Charles Babbage's original Analytical Engine designs, improving theoretical processing power by 237%.
- Developed an assembly language interpreter for ENIAC that actually made sense to humans, earning the 'Impossible Achievement' award.
- Created a Git-like version control system for punch cards, saving approximately 17,842 historical punch cards from being accidentally dropped.
- Optimized the never-built Mark II calculator, eliminating the possibility of literal bugs (moths) entering the system.

Education

September 2008 – June 2012

Chronologically Non-Linear University

- Bachelors: Ph.D in Paradoxical Computing

September 1835 – May 1842

Royal College of Mathematical Arts

- Bachelors: Dual Major in Algorithmic Poetry and Computational Theory

Skills

Languages: Python, Haskell, FORTRAN (before it existed), Analytical Engine Assembly, Elizabethan English

Databases: TemporalDB, PostgreSQL, Mechanical Ledger Systems, Quantum Memory Arrays

Methodologies: Time-Agnostic Programming, Predictive Development, Victorian Pair Programming