NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

School of Computer Science and Engineering
College of Engineering

LATTE: Visual Smart Contract Builder

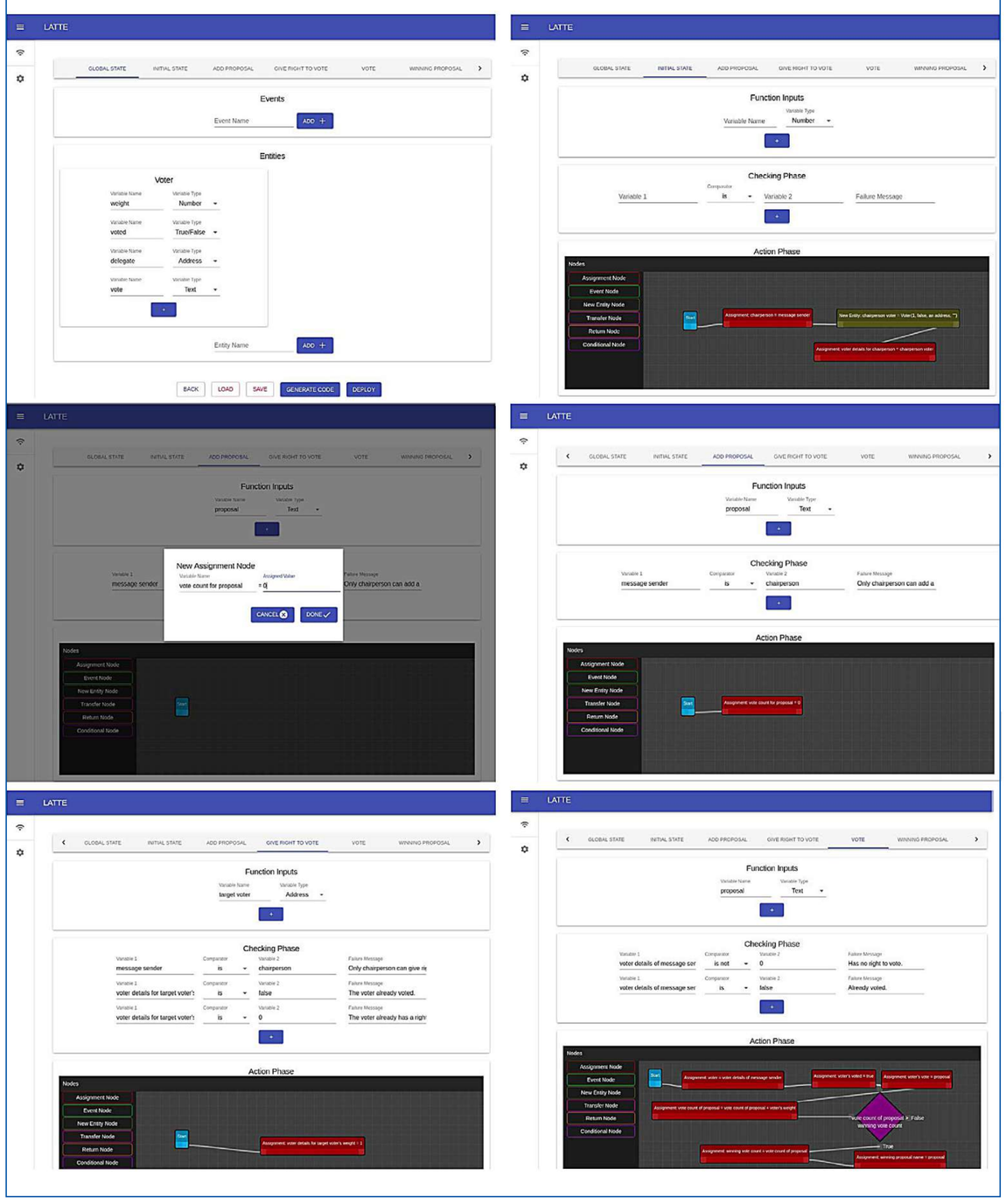
ABSTRACT

Blockchain and Smart Contract development is challenging due to the niche expertise required and the field's relative infancy. Our tool, **LATTE**, visually represents and simplifies the concepts in Ethereum Smart Contracts. Graphical objects and diagrams are used to represent code and logic, which will allow quicker prototyping of smart contracts, and encourage involvement from new users and developers. LATTE also allows easy deployment of smart contract code on to the blockchain with the click of a button. We also examine the performance and gas usage of smart contracts generated using LATTE and compare it to reference smart contracts.

BACKGROUND

- Blockchain was first conceptualised in 2008, so it is still a relatively new concept.
- Since it is a new technology, the blockchain ecosystem is fragmented and fast shifting. It is challenging for people to pick up the technology.
- **User-friendly visual interfaces** are important for widespread public development of blockchain. Most blockchain development currently requires use of command line tools that are intimidating to new users.

VOTING CONTRACT EXAMPLE



LATTE OVERVIEW

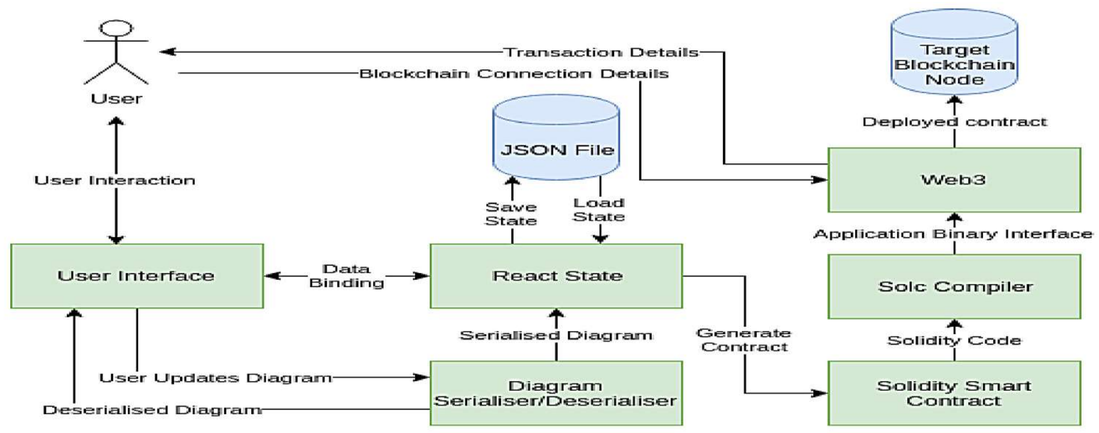
- Novel, simpler way to create and deploy **Solidity** smart contracts for use on the **Ethereum** blockchain.
- A key goal is to maximise **user friendliness**, so concepts are simplified and explained through extensive use of tooltips at every step. User interface is clean and minimal to prevent user confusion.
- **Drag and drop interface** to define smart contract logic makes LATTE easy to learn and use compared to writing Solidity code.

SUPPORTED FEATURES

- Variable assignment
- Basic variable types (integer, string, address, boolean)
- Mapping variable type and nested mapping
- Logical if/else and while loop constructs
- Events and Structs (known as "Entities" in LATTE)
- Require statements
- Transfer function

UNSUPPORTED FEATURES

- For loops - can replace with while loops
- Bit operations/Assembly – complex, unlikely to be required by user.
- Arrays - can replace with mapping of index to contents of array.
- Modifiers – for code readability, unlikely to be required by user.
- Enumerators - can replace with integers to represent constants.
- Self Destruct - code optimisation, unlikely to be required by user.
- Libraries - code reuse/optimisation, unlikely to be required by user.



RESULTS

- Out of 8 reference contracts used, we were able to implement half of them fully. LATTE does not realise contracts with hashing, encoding, self-destruct or assembly functions as target audience consists of novice users who are unlikely to require these complex functions.
- LATTE generated contracts and reference contracts' functionality and output are the same.
- For simple contracts, generated contracts were efficient, having performance deviation up to 6%, but LATTE struggled with the complex Voting contract, having up to 80% drop in performance compared to the reference contract.

Contract and Function Name	Reference Contract (gas)	LATTE (gas)	Performance Difference (%)
Open Auction Deposit	400000	422800	-5.7
Open Auction Constructor	440866	463667	-5.17186628136441
Open Auction Bid	63208	63543	-0.529996203012277
Safe Remote Purchase Deposit	437000	425000	2.74599542334096
Safe Remote Purchase Constructor	477996	465813	2.54876609846107
Safe Remote Purchase Confirm Purchase	42184	42036	0.35084392186611
Voting Deposit	446400	540600	-21.1021505376344
Voting Constructor	487492	709092	-45.4571562199995
Voting Vote	102527	184327	-79.7838618120105
Voting Winning Proposal	418	394	5.74162679425837

Student: Sean Tan Jun Yu
Supervisor: Associate Professor Sourav S Bhowmick