(DEEMED TO BE UNIVERSITY)

BLOCKCHAIN FOR IoT

LAB WORKBOOK

KL University , Green Fields, Vaddeswaram
Andhra Pradesh 522302

LABORATORY WORKBOOK

STUDENT NAME	
REG. NO	
DEPARTMENT	
YEAR	
SEMESTER	
SECTION	
FACULTY	

Table of Contents

Organization of the Student Lab Workbook

The laboratory framework includes a creative element but shifts the time-intensive aspects outside of the Two-Hour closed laboratory period. Within this structure, each laboratory includes three parts: Pre-lab, In-lab, and Post-lab.

Pre-Lab:

The Pre-Lab exercise serves as a bridge between lecture content and the upcoming lab session. It is a preparatory homework assignment designed to take approximately two hours. The objective is to integrate theoretical knowledge from lectures and textbooks into the initial stages of developing a functional software component.

Students enrolled in a two-hour, structured lab session are expected to make a sincere effort to complete the Pre-Lab before attending.

In-Lab:

The In-Lab session is conducted during the scheduled laboratory period and is structured into two key phases.

During the first hour, students address any challenges they encountered while completing the PreLab exercise. This time is dedicated to guided clarification and constructive feedback, enabling students to refine and finalize their Pre-Lab code. The goal is for each student to leave this portion of the session with a functioning implementation—a meaningful step in their learning process.
In the second hour, students work on the In-Lab exercise, which builds upon the Pre-Lab and reinforces key concepts through hands-on application. By the end of the session, students will have received instructor feedback on both their Pre-Lab and In-Lab efforts, ensuring a complete and wellsupported lab experience.

Post-Lab:

The Post-Lab phase is a follow-up homework assignment completed after the laboratory session. It focuses on deepening the student's understanding through analysis and critical reflection. In this phase, students evaluate the performance, efficiency, or practical utility of the concepts or techniques explored during the lab—such as a specific algorithm, method, or system function.

Each Post-Lab exercise is designed to take approximately 120 minutes to complete and encourages students to consolidate their learning by interpreting results, drawing conclusions, and applying their insights to broader contexts.

Software Pre-Requisites:

Python 3.8 or later / MATLAB R2020b or later

`

Experiment-1

Experiment Title: Basics of Blockchain

Aim/Objective:

To understand the basic working of blockchain technology by implementing a simple blockchain structure, adding blocks with transaction data, verifying hash links, and analysing the immutability and security features of blockchain.

Pre-Requisites:

To successfully perform this experiment, students should be familiar with the following:

Theoretical Concepts

- Fundamentals of cryptography
- Concept of a block (data, timestamp, previous hash, current hash)
- Structure of a Blockchain and how blocks are linked together.
-

Pre-Lab:

1. What is the Blockchain?

2. What is Ethereum?

3. What is crypto currency and it's uses?

`

4. Characteristics of Blockchain?

5. Terminology in Blockchain?

In-Lab:

1. Generate and mine a hash of the of any given data using Blockchain demo website. https://andersbrownworth.com/blockchain

Procedure/Program:

Open a Blockchain demo platform

- Go to https://andersbrownworth.com/blockchain (or any other blockchain demo site).

Generate a hash for given data

- Enter sample data into the block's data field (e.g., *"Alice pays Bob 10 coins"*).

`

- Observe how the block's **hash value** changes instantly when the input data is modified.

Mine the block

- Adjust the **Nonce** (number only used once) until the block's hash satisfies the set difficulty requirement (for example, starting with a certain number of leading zeros).
- Observe how mining stabilizes the block by producing a valid hash.

`

Data and Results:

Analysis & Inference:

Viva-Voce Questions (In-Lab):

1. What is a Blockchain?
2. What is a block in Blockchain?
3. Why is the Blockchain considered immutable?
4. What is a Hah value in Blockchain?
5. What is the role of the previous hash in blockchain?

`

Post-Lab:

1. Mine a block in blockchain after change occurred in the block.

Procedure/Program:

1 Open **the Demo Website**

- Visit the Blockchain Demo: https://andersbrownworth.com/blockchain.

2 Observe **the Default Block**

- Each block contains: Block Number, Nonce, Data, Previous Hash, and Hash.
- Initially, the block shows a valid hash (with required leading zeros).

3 **Make a Change in the Block**

- Modify the **data field** (e.g., change *"Alice pays Bob 10 coins"* to *"Alice pays Bob 20 coins"*) OR change the nonce.
- Notice that the block's hash changes immediately and no longer satisfies the difficulty rule (leading zeros).
- The block turns **red/invalid**.

4 **Re-Mine the Block**

- Click on the **"Mine"** button.
- The system automatically adjusts the **nonce** value until a valid hash (with required leading zeros) is found.
- The block turns **green/valid** again.

5 **Check the Blockchain Integrity**

- If the modified block is not the last block, observe that all **subsequent blocks** turn invalid because their previous hash no longer matches.
- Re-mine each invalid block one by one to restore the validity of the entire chain.

6 Finalize

- Once all blocks are re-mined, the blockchain is valid again.

`

- Record the **new nonce** and **hash values** after minin

Data and Results:

Analysis & Inference:

Comment of the Evaluator (if Any)		Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

`

Experiment-2

Experiment Title: Tools and compilers required for a blockchain

Aim/Objective:

To study the essential tools and compilers used in blockchain development. To understand their role in writing, compiling, testing, and deploying smart contracts.

Pre-Requisites:

To successfully complete this experiment, students should have:

Theoretical Concepts

- Basics of Blockchain Concepts
- Fundamentals of smart contracts
- Programming knowledge
- Basic understanding of IDEs and compilers

Pre-Lab:

1. What is Blockchain wallet?

2. What is public key and private key?

3. What is meta mask and uses?

`

4. What is Ganache and it’s uses?

5. What compiler is used to compile and deploy blockchain codes?

In-Lab:

1. Download Meta mask and complete login registration.

Procedure/Program:

1 **Download MetaMask**

- Go to the official website: https://metamask.io/.
- Click **“Download”** and choose your browser extension (Chrome, Firefox, Edge) or mobile app (iOS/Android).
- Add the extension to your browser or install the app.

`

2 **Create a New Wallet**

- Open MetaMask and click **“Get Started” → “Create a Wallet”**.
- Set a strong **password** for your wallet.

3 **Backup Secret Recovery Phrase**

- MetaMask will show a **12-word Secret Recovery Phrase**.
- Write it down safely offline — do **not share it** with anyone.
- Confirm the phrase as instructed to complete the setup.

4 **Login to MetaMask**

- After setup, enter your **password** to unlock the wallet anytime.
- You can now send/receive test Ether and connect MetaMask to blockchain networks like Ganache or Ethereum testnets.

`

Data and Results:

Analysis & Inference:

Viva-Voce Questions (In-Lab):

1. What are blockchain development tools?
2. Name any two commonly used blockchain IDEs.
3. What is the role of the Solidity compiler?
4. What is Remix IDE used for?
5. what is Ethereum?

`

Post-Lab:

1. Installation of Ganache.

Procedure/Program:

1. **Download Ganache**
   - Go to the official website: https://trufflesuite.com/ganache/.
   - Click **Download** and choose the version for your operating system (Windows, macOS, Linux).
2. **Install Ganache**
   - Run the downloaded installer.
   - Follow the on-screen instructions and complete the installation.
3. **Launch Ganache**
   - Open Ganache after installation.
   - A local blockchain workspace will be created automatically with pre-funded accounts.
4. **Explore Accounts and Settings**
   - Observe the generated accounts, private keys, and balances.
   - Ganache is now ready to connect with **Remix IDE** or other blockchain development tools for testing smart contracts.

`

Data and Results:

Analysis & Inference:

Comment of the Evaluator (if Any)		Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

`

Experiment-3

Experiment Title: Write a smart contract in Remix IDE .

Aim/Objective:

To learn how to create, compile, and deploy a simple smart contract using **Solidity** in **Remix IDE**.

Pre-Requisites:

To successfully complete this experiment, students should:

Theoretical Concepts

1. Basic understanding of **Blockchain concepts** (blocks, transactions, contracts).
2. Knowledge of **Ethereum platform** and its use of smart contracts.
3. Familiarity with **Solidity programming language** syntax and structure.
4. Understanding of **Integrated Development Environments (IDEs)**.
5. Internet access and a web browser to use **Remix IDE** (online Solidity editor and compiler).

Pre-Lab:

1. Which programming is used to write a smart contract?

2. What is solidity and why it's more important in blockchain technology?

`

3. What are the basic datatypes in blockchain?

4. What is remix IDE?

5. What is ethereum?

`

In-Lab:

1. Compile and deploy a basic solidity program using remix ide.

Procedure/Program:

1 Open **Remix IDE**

- Go to https://remix.ethereum.org in your web browser.

2 Create **a New Solidity File**

- In the **File Explorer**, click **"+"** to create a new file (e.g., SimpleStorage.sol).
- Write a basic Solidity contract

3 Compile **the Contract**

- Go to the **Solidity Compiler** tab.
- Select the correct compiler version (matching pragma).
- Click **"Compile SimpleStorage.sol"** and ensure there are no errors.

4 Deploy **the Contract**

- Go to the **Deploy & Run Transactions** tab.
- Choose **JavaScript VM** (default, for testing).
- Click **Deploy**.
- After deployment, the contract will appear under **Deployed Contracts**.

5 Interact **with the Contract**

- Use the displayed buttons to call functions:
  - set() → store a value.
  - get() → retrieve the stored value.
- Observe the transaction results in the Remix console.

`

Data and Results:

Analysis & Inference:

Viva-Voce Questions (In-Lab):

1. What is a smart contract?
2. Which programming language is mainly used to write smart contracts in Ethereum?
3. What is Remix IDE?
4. What are the main components of a Solidity smart contract?

`

Post-Lab:

1.Compile and deploy a basic math program in Remix IDE.

Procedure/Program:

1 **Open Remix IDE**

- Go to https://remix.ethereum.org in your web browser.

2 **Create a New Solidity File**

- In the **File Explorer**, click **"+"** to create a new file (e.g., MathOperations.sol).
- Write a simple Solidity contract for basic math operations:

3 **Compile the Contract**

- Go to the **Solidity Compiler** tab.
- Select the correct compiler version.
- Click **"Compile MathOperations.sol"** and ensure there are no errors.

4 **Deploy the Contract**

- Go to the **Deploy & Run Transactions** tab.
- Select **JavaScript VM** as the environment for testing.
- Click **Deploy**.
- The contract will appear under **Deployed Contracts**.

5 **Interact with the Contract**

- Click on the functions (add, subtract, multiply, divide) and enter values to perform operations.
- Observe the results and verify correctness in the Remix console.

`

Data and Results:

Analysis & Inference:

Comment of the Evaluator (if Any)		Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

`

Experiment 4

Experiment Title: **Creating attesting environment in Blockchain**

Aim/Objective:

To understand the concept of **attestation** in blockchain, where identities, data, or transactions are verified and validated by trusted parties. To create a simple attesting environment that demonstrates how blockchain ensures **authenticity, integrity, and trust** without relying on a centralized authority.

Pre-Requisites:

To successfully complete this experiment, students should:

Theoretical Concepts

- Basic knowledge of **Blockchain concepts** (blocks, hash, transactions, consensus).
- Understanding of **public and private keys** in cryptography.
- Familiarity with **digital signatures** and their role in authentication.
- Awareness of **identity verification** and why trust is needed in decentralized systems.
- Basic experience with **smart contracts** or blockchain demo tools.

Pre-Lab:

1. How to create and login into meta mask?

2. How to download Ganache?

`

3. How connect meta mask and Ganache?

4. How to get testing currency in meta mask?

5. What is Ganache and it’s uses ?

`

In-Lab:

1. Configuring network in meta mask with ganache.

Procedure/Program:

1. **Open Ganache**
   - Launch Ganache and note the **RPC server URL** (usually http://127.0.0.1:7545).
   - Copy the **chain ID** displayed in Ganache.
2. **Open MetaMask**
   - Click the **MetaMask** browser extension.
   - Unlock your wallet with your password.
3. **Add a Custom Network**
   - Click on the **network dropdown** at the top.
   - Select **"Add Network"** or **"Add Custom RPC"**.
   - Enter the following details:
     - **Network Name:** Ganache Local
     - **New RPC URL:** http://127.0.0.1:7545
     - **Chain ID:** (copy from Ganache, usually 1337)
     - **Currency Symbol:** ETH (optional)
     - **Block Explorer URL:** leave blank (optional)
4. **Save and Connect**
   - Click **Save**.
   - MetaMask is now connected to the Ganache local blockchain.
5. **Import Accounts (Optional)**
   - Copy a **private key** from Ganache and import it in MetaMask to use test accounts.

`

Data and Results:

Analysis & Inference:

Viva-Voce Questions (In-Lab):

1. What is a testing environment in blockchain?
2. Why do we need a testing environment before deploying real contracts?
3. Name one tool used for blockchain testing.
4. What is Ganache used for?
5. Why is it safe to use fake Ether in testing?

`

Post-Lab:

1. Connect meta mask and ganache using Remix IDE

Procedure/Program:

1 Set **Up Ganache**

- Open **Ganache** and note the **RPC server URL** (usually http://127.0.0.1:7545) and **chain ID** (usually 1337).

2 Configure **MetaMask**

- Open **MetaMask** and unlock your wallet.
- Add a **custom network** with the following:
    - Network Name: Ganache Local
    - RPC URL: http://127.0.0.1:7545
    - Chain ID: 1337
    - Currency Symbol: ETH (optional)
- Save the network.

3 Import **Ganache Account into MetaMask**

- Copy a **private key** from Ganache and import it into MetaMask to access test accounts.

4 **Open Remix IDE**

- Go to https://remix.ethereum.org.

5 **Select Environment in Remix**

- Go to **Deploy & Run Transactions** tab.
- Set **Environment** to **Injected Web3**.
- Remix will connect through MetaMask, which is linked to Ganache.

6 **Deploy Contract**

- Compile your smart contract.
- Click **Deploy**.
- Confirm the transaction in MetaMask.
- The contract is now deployed on the Ganache local blockchain.

`

Data and Results:

Analysis & Inference:

Comment of the Evaluator (if Any)		Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

`

Experiment-5

Experiment Title: **Make your first contract using Remix ide, Ganache, MetaMask**

Aim/Objective:

To create, compile, and deploy a simple smart contract using **Remix IDE**. To interact with the contract using **Ganache** (local blockchain) and **MetaMask** (wallet), understanding the end-to-end deployment workflow.

Pre-Requisites:

To successfully complete this experiment, students should:

Theoretical Knowledge

1. Basic understanding of **blockchain concepts** (blocks, transactions, smart contracts).
2. Knowledge of **Solidity programming** basics.
3. Familiarity with **Remix IDE**, **Ganache**, and **MetaMask**.
4. Basic knowledge of **Ethereum test networks** and fake Ether usage.

Pre-Lab:

1. What is transaction?

2. Which algorithm used in transcarion?

`

3. What is encryption and decryption?

4. What is SHA256 and uses of it?

5. How to do transaction in Remix IDE?

`

In-Lab:

1. Adding fake currency to meta mask from ganache private key.

Procedure/Program:

1. **Open Ganache**
   - Launch Ganache and view the **pre-funded accounts**.
   - Copy the **private key** of one of the accounts.
2. **Open MetaMask**
   - Unlock your MetaMask wallet.
   - Switch to the **Ganache Local Network** (custom network configured earlier).
3. **Import Account**
   - Click on the account icon → **Import Account**.
   - Paste the **private key** copied from Ganache.
   - Click **Import**.
4. **Verify Balance**
   - The account will now show the **fake Ether balance** from Ganache.
   - You can use this fake Ether to deploy contracts and test transactions safely.

`

Data and Results:

Analysis & Inference:

Viva-Voce Questions (In-Lab):

1. What is a smart contract?
2. What is Remix IDE used for?
3. Why do we need Ganache in development?
4. What is the role of MetaMask?

`

Post-Lab:

1.Make a transaction in Remix IDE using meta mask and Ganache.

Procedure/Program:

1 **Set Up Ganache and MetaMask**

- Open **Ganache** and note the RPC URL (http://127.0.0.1:7545) and chain ID (1337).
- In **MetaMask**, switch to the **Ganache Local Network**.
- Import a Ganache account using its **private key** to access fake Ether.

2 Open **Remix IDE**

- Go to https://remix.ethereum.org.
- Create a new Solidity file with a simple contract

3 **Compile the Contract**

- Go to the **Solidity Compiler** tab.
- Select the correct compiler version and click **Compile**.

4 **Deploy the Contract**

- Go to **Deploy & Run Transactions** tab.
- Set **Environment** to **Injected Web3** (MetaMask connection).
- Click **Deploy** and confirm the transaction in MetaMask.

5 **Make a Transaction**

- After deployment, use the set() function to send a value to the contract.
- Confirm the transaction in MetaMask.
- Check the updated value using the get() function.

6 **Verify on Ganache**

- Observe the transaction details, gas used, and updated balance in Ganache.

`

Data and Results:

Analysis & Inference:

Comment of the Evaluator (if Any)		Evaluator's Observation Marks Secured:_____out of _____ Full Name of the Evaluator: Signature of the Evaluator Date of Evaluation:

`