FULL STACK FRAMEWORKS LAB MANUAL

STUDENT INFORMATION

Field	Details
Roll Number	CB.SC.U4CSE23720
Student Name	Guru.D
Course Code	23CSE461
Course Name	Full Stack Frameworks
GitHub Profile	https://github.com/Guru006-Dev
Project Repository	https://github.com/Guru006-Dev/react-math-assignments
Live Application	https://react-math-assignments.vercel.app/
Semester/Year	2026

COURSE INFORMATION

Category	Details
Course Code	23CSE461
Course Name	FULL STACK FRAMEWORKS
Department	Computer Science and Engineering
Credits	Lab Course

Course Objectives

No.	Objective
1	Understand how web development has become easier with the introduction of frameworks
2	Learn full stack web development principles and practices
3	Develop, optimize and maintain websites using full-stack frameworks
4	Master important full stack frameworks for modern web development

Course Outcomes

Code	Outcome Description
CO1	Learn how to develop single page applications (SPAs) efficiently using front-end framework
CO2	Learn to use backend frameworks to develop web and mobile applications robustly
CO3	Learn to build highly available and scalable internet applications using document databases
CO4	Design and develop full stack web projects using front-end, back-end and database frameworks

---

SYLLABUS

Unit I - React JS

Creating and using components, bindings, props, states, events, Working with components, Conditional rendering, Building forms, Getting data from RESTful APIs, Routing, CRUD with Firebase, Redux, React and Redux, Function vs. class based components, Hooks.

Unit II - Express JS

Node JS – Basics, setup, console, command utilities, modules, events, Express JS – Routing, HTTP methods, CSS, Bootstrap, JavaScript, React, Redux, Node, Express, URL building, Templates, Static files, Form data, Database, Cookies, Sessions, Authentication, RESTful APIs, Scaffolding, Error handling, Debugging.

Unit III - Mongo DB

Mongo DB ecosystem, Importing and Exporting data, Mongo query language, Updating documents, Aggregation framework, System and user generated variables, Scheme validation, Data modelling, Indexing, Performance.

REFERENCE MATERIALS

Technology	Documentation URL
ReactJS	https://react.dev/
NodeJS	https://nodejs.org/docs/
MongoDB	https://www.mongodb.com/docs/
ExpressJS	https://expressjs.com/
JavaScript	https://developer.mozilla.org/en-US/docs/Web/JavaScript
HTML	https://developer.mozilla.org/en-US/docs/Web/HTML
Responsive HTML	https://web.dev/responsive-web-design-basics/
CSS	https://developer.mozilla.org/en-US/docs/Web/CSS

LIST OF EXPERIMENTS

Sl.No	Ex.No	Date	Title of the Experiments	Page No.
1	1		Mathematical Operations (Factorial, Fibonacci, Prime) - ReactJS	7
2	2		Sum of Digits Calculator - ReactJS	9
3	3		Question Paper Set Selector (Class & Function) - ReactJS	11
4	4		Basic Calculator - ReactJS	13
5	5		Kids Calculator Game - ReactJS	15

EX.NO: 1 - MATHEMATICAL OPERATIONS (FACTORIAL, FIBONACCI, PRIME)

AIM

Create a ReactJS application to perform three mathematical operations: - Calculate factorial of a number - Generate Fibonacci series - Check if a number is prime

GITHUB REPOSITORY

https://github.com/Guru006-Dev/react-math-assignments

DEPLOYED URL

https://react-math-assignments.vercel.app/question-a

PROJECT LOCATION

```
c:\Users\Guru\Desktop\Full Stack\React_project
```

COMPONENT FILE

```
src/components/QuestionA.jsx
```

LIST OF FILE NAMES WITH PURPOSE

FileName	Purpose
QuestionA.jsx	Main component with all three mathematical operations
App.jsx	Routing configuration
index.css	Global styling

CONCEPTS USED IN THE APPLICATION

Concept Name	General Purpose	Code File Where Used
React useState Hook	Managing input and results state	QuestionA.jsx
Factorial Algorithm	Iterative multiplication for n!	QuestionA.jsx
Fibonacci Algorithm	Generating sequence using iteration	QuestionA.jsx
Prime Check Algorithm	Checking divisibility up to √n	QuestionA.jsx
Multiple Function Calls	Executing all three operations together	QuestionA.jsx

ALGORITHMS

Factorial:

```
1. If n < 0: return invalid
2. If n = 0 or 1: return 1
3. result = 1
4. For i from 2 to n: result *= i
5. Return result
```

Fibonacci:

```
1. Initialize array with [0, 1]
2. For i from 2 to n:
3.   fib[i] = fib[i-1] + fib[i-2]
4. Return array
```

Prime Check:

```
1. If n < 2: return false
2. If n = 2: return true
3. If n is even: return false
4. For i from 3 to √n (step 2):
5.   If n % i = 0: return false
6. Return true
```

KEY FEATURES

- Single input generates all three results
- Efficient algorithms for each operation
- Clear result display with labels
- Handles edge cases (negative, zero, etc.)

OUTPUT FORMAT

Operation	Output Display
Factorial	n! = result (e.g., 5! = 120)
Fibonacci	First n terms as comma-separated values
Prime Check	Boolean result with ✓ (Prime) or ✗ (Not Prime)

TEST CASES

Input	Factorial	Fibonacci (First n)	Is Prime?
5	120	0, 1, 1, 2, 3	✓ Yes
7	5040	0, 1, 1, 2, 3, 5, 8	✓ Yes
10	3628800	0, 1, 1, 2, 3, 5, 8, 13, 21, 34	✗ No

RESULT

Successfully implemented a comprehensive mathematical operations component in ReactJS demonstrating factorial, Fibonacci, and prime number algorithms.

EX.NO: 2 - SUM OF DIGITS CALCULATOR

AIM

Read a number and provide the sum of all its digits.

GITHUB REPOSITORY

https://github.com/Guru006-Dev/react-math-assignments

DEPLOYED URL

https://react-math-assignments.vercel.app/question-b

PROJECT LOCATION

```
c:\Users\Guru\Desktop\Full Stack\React_project
```

COMPONENT FILE

```
src/components/QuestionB.jsx
```

LIST OF FILE NAMES WITH PURPOSE

FileName	Purpose
QuestionB.jsx	Component for sum of digits calculation
App.jsx	Routing configuration
index.css	Global styling

CONCEPTS USED IN THE APPLICATION

Concept Name	General Purpose	Code File Where Used
React useState Hook	Managing input number and result	QuestionB.jsx
String Manipulation	Converting number to string for digit extraction	QuestionB.jsx
Array Methods	Iterating through digits	QuestionB.jsx
parseInt() Function	Converting string digits to numbers	QuestionB.jsx
Math.abs()	Handling negative numbers	QuestionB.jsx

ALGORITHM

```
1. Accept number from user
2. Convert to absolute value (handle negatives)
3. Convert number to string
4. Initialize sum = 0
5. For each character in string:
6.   Convert to integer
7.   Add to sum
8. Display individual digits and sum
```

EXAMPLE CALCULATION

```
Input: 12345
Digits: 1, 2, 3, 4, 5
Calculation: 1 + 2 + 3 + 4 + 5 = 15
Result: 15
```

KEY FEATURES

Feature	Description
Input Support	Accepts any integer (positive or negative)
Digit Breakdown	Shows individual digits separated by +
Formula Display	Visual calculation formula
Examples	Reference cards with sample calculations
Visual Addition	Clear step-by-step addition representation

TEST CASES

Input Number	Digits	Sum
123	1 + 2 + 3	6
9876	9 + 8 + 7 + 6	30
555	5 + 5 + 5	15
12345	1 + 2 + 3 + 4 + 5	15
-456	4 + 5 + 6	15

OUTPUT FORMAT

```
Original Number: 12345
Digits: 1 + 2 + 3 + 4 + 5
Sum of Digits: 15
Calculation: 1 + 2 + 3 + 4 + 5 = 15
```

RESULT

Successfully created a sum of digits calculator with visual breakdown and clear presentation of the calculation process.

EX.NO: 3 - QUESTION PAPER SET SELECTOR

AIM

Create a ReactJS application to determine question paper set based on roll number: - Odd roll number → Set 1 - Even roll number → Set 2

Implement using both **Class Component** and **Function Component**.

GITHUB REPOSITORY

https://github.com/Guru006-Dev/react-math-assignments

DEPLOYED URL

https://react-math-assignments.vercel.app/question-c

PROJECT LOCATION

`c:\Users\Guru\Desktop\Full Stack\React_project`

COMPONENT FILE

`src/components/QuestionC.jsx`

LIST OF FILE NAMES WITH PURPOSE

FileName	Purpose
QuestionC.jsx	Main component containing both implementations
App.jsx	Routing configuration
index.css	Global styling

CONCEPTS USED IN THE APPLICATION

Concept Name	General Purpose	Code File Where Used
React Function Component	Modern approach with hooks	QuestionC.jsx (QuestionCFunction)
React Class Component	Traditional approach with class syntax	QuestionC.jsx (QuestionCClass)
useState Hook	State management in function component	QuestionCFunction
this.state	State management in class component	QuestionCClass
Modulo Operator (%)	Determining odd/even	Both components
Event Handling	Button clicks and input changes	Both components

ALGORITHM

```
1. Accept roll number from user
2. Validate input (must be positive integer)
3. Calculate rollNumber % 2
4. If result = 0: Set = 2 (Even)
5. If result = 1: Set = 1 (Odd)
6. Display assigned set
```

FUNCTION COMPONENT IMPLEMENTATION

```
function QuestionCFunction() {
  const [rollNumber, setRollNumber] = useState('');
  const [result, setResult] = useState(null);

  const determineSet = () => {
    const roll = parseInt(rollNumber);
    const set = roll % 2 === 0 ? 2 : 1;
    setResult({ rollNumber: roll, set, isEven: roll % 2 === 0 });
  };
}
```

CLASS COMPONENT IMPLEMENTATION

```
class QuestionCClass extends React.Component {
  constructor(props) {
    super(props);
    this.state = { rollNumber: '', result: null };
  }

  determineSet = () => {
    const roll = parseInt(this.state.rollNumber);
    const set = roll % 2 === 0 ? 2 : 1;
    this.setState({ result: { rollNumber: roll, set, isEven: roll % 2 ===
  }
}
```

KEY FEATURES

Feature	Description
Dual Implementation	Side-by-side comparison of function and class components
Identical Logic	Same functionality with different syntax approaches
Visual Distinction	Different color schemes for each component type
Set Assignment	Clear display of assigned question paper set
Input Validation	Ensures valid roll number entry

TEST CASES

Roll Number	Type	Assigned Set
1	Odd	Set 1
2	Even	Set 2
7	Odd	Set 1
10	Even	Set 2
23	Odd	Set 1
48	Even	Set 2

COMPARISON: FUNCTION VS CLASS COMPONENTS

Aspect	Function Component	Class Component
Syntax	Simple, concise	More verbose
State Management	useState Hook	this.state, this.setState()
Lifecycle	useEffect Hook	componentDidMount, etc.
this Binding	□ Not needed	□ Required for methods
Code Length	Shorter	Longer
Modern Standard	□ Recommended (2023+)	□□ Legacy support
Performance	Slightly Better	Standard
Learning Curve	Easier	Moderate

RESULT

Successfully demonstrated both function and class component implementations for the same functionality, highlighting the evolution of React development patterns.

EX.NO: 4 - BASIC CALCULATOR

AIM

Create a fully functional calculator program using ReactJS with all basic arithmetic operations.

GITHUB REPOSITORY

https://github.com/Guru006-Dev/react-math-assignments

DEPLOYED URL

https://react-math-assignments.vercel.app/question-d

PROJECT LOCATION

```
c:\Users\Guru\Desktop\Full Stack\React_project
```

COMPONENT FILE

```
src/components/QuestionD.jsx
```

LIST OF FILE NAMES WITH PURPOSE

FileName	Purpose
QuestionD.jsx	Calculator component with all operations
App.jsx	Routing configuration
index.css	Calculator grid and button styling

CONCEPTS USED IN THE APPLICATION

Concept Name	General Purpose	Code File Where Used
React useState Hook	Managing calculator state and operations	QuestionD.jsx
Multiple State Variables	Display, previousValue, operation, waitingForOperand	QuestionD.jsx
Event Handling	Button click handlers	QuestionD.jsx
CSS Grid Layout	Calculator button layout	index.css
Switch Statement	Operation execution	QuestionD.jsx

STATE MANAGEMENT

```
const [display, setDisplay] = useState('0');
const [previousValue, setPreviousValue] = useState(null);
const [operation, setOperation] = useState(null);
const [waitingForOperand, setWaitingForOperand] = useState(false);
```

CALCULATOR OPERATIONS

Operation	Symbol	Function
Addition	+	a + b
Subtraction	−	a - b
Multiplication	×	a * b
Division	÷	a / b
Modulo	%	a % b
Sign Toggle	+/-	value * -1
Clear	AC	Reset all
Decimal	.	Add decimal point

ALGORITHM

```
1. User inputs first number via digit buttons
2. User selects operation (+, -, ×, ÷, %)
3. Store first value and operation
4. User inputs second number
5. On pressing '=', calculate result
6. Display result
7. Support chaining operations
```

CALCULATOR LAYOUT

```
┌─────────────────────────────┐
│         Display (0)         │
├───────┬───────┬───────┬─────┤
│  AC   │  +/-  │   %   │  ÷  │
├───────┼───────┼───────┼─────┤
│  7    │   8   │   9   │  ×  │
├───────┼───────┼───────┼─────┤
│  4    │   5   │   6   │  -  │
├───────┼───────┼───────┼─────┤
│  1    │   2   │   3   │  +  │
├───────┴───────┼───────┼─────┤
│       0       │   .   │  =  │
└───────────────┴───────┴─────┘
```

KEY FEATURES

Category	Features
Operations	Addition, Subtraction, Multiplication, Division, Modulo
Number Support	Integers, Decimals, Negative numbers
Special Functions	Clear (AC), Sign Toggle (+/-)
Advanced	Chain calculations, Operation chaining
UI/UX	Responsive grid layout, Professional design

TEST CASES

Calculation	Expected Result
5 + 3	8
10 - 7	3
6 × 4	24
15 ÷ 3	5
17 % 5	2
2.5 + 3.7	6.2
8 ÷ 0	Error (prevented)

EDGE CASES HANDLED

Case	Handling
Division by Zero	□ Prevents operation, returns 0
Multiple Decimals	□ Ignores additional decimal points
Operation Chaining	□ Continues calculation with result
Display Overflow	□ Handles large numbers
Invalid Input	□ Resets to valid state

RESULT

Successfully implemented a fully functional calculator in ReactJS with professional UI and comprehensive operation support.

EX.NO: 5 - KIDS CALCULATOR GAME

AIM

Create a calculator program with the addition of game concepts for kids, making math learning fun and engaging through gamification.

GITHUB REPOSITORY

https://github.com/Guru006-Dev/react-math-assignments

DEPLOYED URL

https://react-math-assignments.vercel.app/question-ef

PROJECT LOCATION

`c:\Users\Guru\Desktop\Full Stack\React_project`

COMPONENT FILE

`src/components/QuestionEF.jsx`

LIST OF FILE NAMES WITH PURPOSE

FileName	Purpose
QuestionEF.jsx	Kids calculator game component
App.jsx	Routing configuration
index.css	Game-specific styling and animations

CONCEPTS USED IN THE APPLICATION

Concept Name	General Purpose	Code File Where Used
React useState Hook	Managing calculator and game state	QuestionEF.jsx
Game State Management	Score, streak, total calculations	QuestionEF.jsx
useEffect Hook	Handling celebration animations	QuestionEF.jsx
Conditional Rendering	Dynamic emoji and messages	QuestionEF.jsx
CSS Animations	Pulse and fade effects	index.css
Random Selection	Encouragement messages	QuestionEF.jsx

GAME STATE VARIABLES

```
const [score, setScore] = useState(0);
const [streak, setStreak] = useState(0);
const [totalCalculations, setTotalCalculations] = useState(0);
const [showCelebration, setShowCelebration] = useState(false);
const [currentEmoji, setCurrentEmoji] = useState('□');
```

GAMIFICATION ELEMENTS

1. Score System

- Points awarded for each calculation

- Formula: `points = floor(abs(result) / 10) + 10`
- Cumulative score tracking

2. Streak Counter

- Increments with each successful calculation
- Resets on game reset
- Triggers emoji upgrades

3. Emoji Progression

Streak Level	Emoji	Description
Start	□	Initial state
1-3	□	Correct calculation
4-6	□	Milestone achieved
7-9	□	Super performance
10+	□	Genius level

4. Encouragement Messages

Random selection from: - “Great job!” - “You’re a math star!” - “Amazing!” - “Keep going!” - “Fantastic!” - “You’re on fire!” - “Brilliant!” - “Superb!”

ALGORITHM

```
1. User performs a calculation
2. On pressing '=':
   a. Calculate result
   b. Award points based on result magnitude
   c. Increment streak counter
   d. Update total calculations
   e. Determine emoji based on streak
   f. Show random encouragement
   g. Trigger celebration animation
3. Display updated score, streak, total
4. Reset game option available
```

SCORE CALCULATION EXAMPLE

```
Calculation: 50 + 30 = 80
Points = floor(80 / 10) + 10 = 8 + 10 = 18 points
New Score = Previous Score + 18
Streak = Streak + 1
```

KEY FEATURES

Category	Features
Calculator	All arithmetic operations, Decimal support, Clear & Reset
Scoring	Real-time score tracking, Points based on calculation magnitude
Progression	Streak counter, Progressive emoji rewards, Total calculations
Feedback	Celebration animations, Random encouragement messages
Education	Fun math practice, Positive reinforcement, Confidence building

EDUCATIONAL BENEFITS

Benefit	How It Helps
Engagement	Makes math practice fun and interactive
Motivation	Positive reinforcement through rewards
Progress	Visual tracking of improvements
Practice	Encourages repeated calculations
Confidence	Builds self-esteem through achievements

GAME MECHANICS SUMMARY

Mechanic	Purpose	Implementation
Score	Track performance	+10-50 points per calculation
Streak	Encourage consistency	Increments with each = press
Emojis	Visual rewards	Changes based on streak level
Messages	Positive feedback	Random selection from 8 options
Animation	Celebration	1.5s pulse effect

GAME INTERFACE LAYOUT

```
┌──────────────────────────────────┐
│  Score: 120  │ Streak: 5□ │ Total: 8  │
├──────────────────────────────────┤
│          □ (Emoji)          │
│      "You're Amazing!"        │
├──────────────────────────────────┤
│      Calculator Display        │
│     [Calculator Buttons]       │
│     [AC] [Reset Game] [÷]      │
│      [7] [8] [9] [×]          │
│      [4] [5] [6] [-]          │
│      [1] [2] [3] [+]          │
│      [  0   ] [.] [=□]       │
└──────────────────────────────────┘
```

CELEBRATION ANIMATION

Property	Value
Trigger	On calculation completion (= button)
Duration	1.5 seconds
Effect	Pulse animation on emoji
Message	Random encouragement phrase
Transition	Smooth fade-in / fade-out

RESULT

Successfully created an educational kids calculator game that combines mathematical operations with engaging game mechanics, making learning fun and rewarding for children.

CONCLUSION

This lab manual documents **5 comprehensive ReactJS experiments** demonstrating advanced React concepts and practical applications:

ReactJS Experiments (All 5)

1. Mathematical operations (Factorial, Fibonacci, Prime checking)
2. Sum of digits calculator
3. Question paper selector (Function & Class components)
4. Basic calculator
5. Kids calculator game (with gamification)

Skills Demonstrated

□ **React Concepts:** Components, Hooks, State Management, Props, Event Handling
□ **Component Types:** Both Function Components (modern) and Class Components (traditional)
□ **JavaScript:** ES6+ syntax, Array methods, Mathematical algorithms
□ **CSS:** Responsive design, Animations, Glassmorphism, Grid layouts
□ **Problem Solving:** Algorithms for factorial, Fibonacci, prime checking
□ **UI/UX Design:** Modern interfaces, User experience, Gamification
□ **Deployment:** GitHub version control, Vercel hosting

Technologies Mastered

- React 18.2 with Hooks
- React Router for navigation
- Vite build tool
- Modern CSS3 with animations
- Component architecture
- State management patterns

Project Deployment

- **GitHub Repository:** https://github.com/Guru006-Dev/react-math-assignments
- **Live Application:** https://react-math-assignments.vercel.app/
- All components deployed and accessible online

Submitted By:
Name: Guru.D
Roll No: CB.SC.U4CSE23720
Course: Full Stack Frameworks (23CSE461)
GitHub: https://github.com/Guru006-Dev

Date: January 2026

Instructor Signature: _______________