SALESFORCE DEVELOPER

PROJECT TITTLE : WORKFORCE ADMINISTRATION SOLUTION

TEAM ID : NM2023TMID0250

TEAM MEMBERS

PAVITHRA M	**(611420104054)**
DHIVYABHARATHI A	**(611420104018)**
JAYASHREE T	**(611420104026)**
NANDHINEESWARI D	**(611420104047)**

COLLEGE SPOC	**FACULTY MENTOR**
MANIMEGALAI M , M.E	**KOWSALYA N , M.E**

PROJECT DEVELOPMENT PHASE :

CODE LAYOUT,READABILITY AND REUSABILITY

Consistent Code Style:

Enforce a consistent code style throughout the project. Use coding standards and style guides that are agreed upon by your development team.

Meaningful Variable and Function Names:

Use descriptive and meaningful names for variables, functions, and classes to make the code self-explanatory.

Comments and Documentation:

Include comments to explain complex logic, especially if it's not immediately obvious. Provide clear and up-to-date documentation for APIs and important functions.

Modularization:

Break down the code into modular components or functions with specific responsibilities. This promotes code reusability and easier testing.

Indentation and Formatting:

Consistently use proper indentation and formatting. Tools like linters and formatters can help maintain a consistent code layout.

Avoid Magic Numbers and Strings:

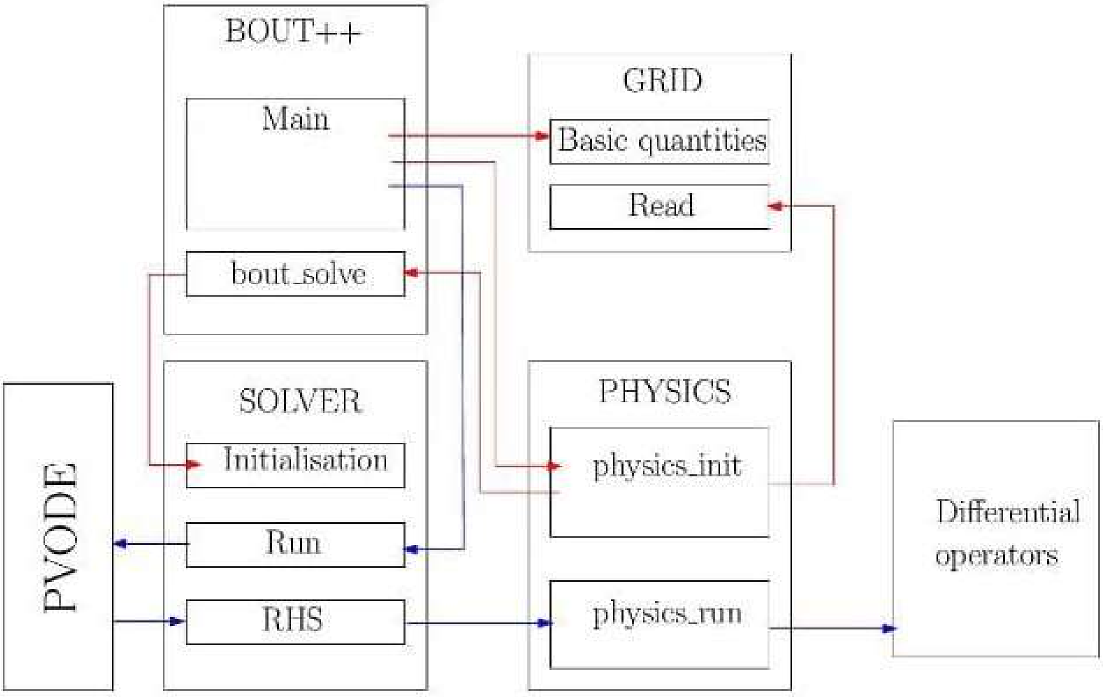
Replace magic numbers and strings with named constants or configuration variables to enhance code maintainability.

Error Handling:

Implement effective error handling to provide clear error messages and gracefully handle exceptions.

Version Control:

Use version control systems like Git to track changes, collaborate with a team, and easily roll back to previous code versions if necessary.



Testing:

Write unit tests to ensure the correctness of your code. Well-tested code is more maintainable and less error-prone.

Code Reviews:

Regularly conduct code reviews within your development team to identify issues, suggest improvements, and maintain code quality.

DRY (Don't Repeat Yourself) Principle:

Avoid duplicating code. Repeated logic should be placed in functions or classes to reduce redundancy.

Use Design Patterns:

Apply well-known design patterns to solve common problems. This can make your code more understandable to other developers familiar with these patterns.

Dependency Management:

Use package managers and dependency injection to manage external libraries and components.

Performance Optimization:

Optimize code for performance when necessary, but not at the expense of readability. Balance between clarity and efficiency.