1. Introduction to ERP and Frappe/ERPNext

1.1 What is ERP?

- Definition, core modules (Accounting, Inventory, HR, etc.).
- Benefits of ERP for businesses.

1.2 Overview of ERPNext

- What is ERPNext?
- Key Features: Accounting, CRM, HR, Manufacturing, etc.
- Use cases: SMEs, manufacturing, retail, NGOs, etc.

1.3 Overview of Frappe Framework

- Open-source, Python-based, full-stack framework.
- Relationship with ERPNext.
- Key features: Database abstraction, REST API, role-based permissions.

1.4 Setup Types

- **Local Setup** : Prerequisites (Python, Node.js, MariaDB/MySQL).
    - Step-by-step installation on Windows (WSL2), macOS, and Linux.
- **Cloud Setup** : Deploying ERPNext on cloud platforms (AWS, DigitalOcean, etc.).
- **Docker Setup** : Using Docker for quick and isolated environments.

1.5 DocTypes

- **Types** : Single, Child, Submittable.
- Naming conventions and metadata.
- Creating custom DocTypes.

1.6 Module Definition

- Linking DocTypes to modules.
- Organizing apps and modules.

1.7 Workflows

- Creating workflows.
- States and transitions.
- Use cases: Approval processes, task management.

1.8 Views

- List View: Filtering, sorting, and bulk actions.
- Form View: Editing records and managing fields.
- Report View: Generating dynamic reports.

1.9 Scripting

- Client Scripts: JavaScript for frontend logic.
- Server Scripts: Python for backend logic.
- Examples: Field validation, auto-fill, conditional visibility.

1.10 Fields

- Adding fields to DocTypes.

- Field types and options (Data, Link, Table, etc.).
- **Field Dependencies** : Conditional fields, dependent dropdowns, and dynamic options.

2. Core Concepts of Frappe Framework

2.1 Architecture

- MVC (Model-View-Controller) pattern.
- Client-server communication (Socket.IO, REST API).

2.2 Key Components

- Models: Fields, permissions, and validation.
- Controllers: Server-side scripting (Python).
- Database Layer: MariaDB/PostgreSQL integration.

2.3 Frappe's Desk Interface

- Workspace, forms, lists, and dashboards.
- User permissions and roles.
- **Custom Navigation Menus** : Adding custom menu items and organizing navigation.

2.4 Developer Tools and Resources

- Frappe Desk: Debugging and error tracking.
- Developer Mode: Enabling and using it effectively.
- Useful resources: Official docs, forums, GitHub repos.

3. Automation and Workflows

- **What is a Workflow?**
    - Definition and purpose of workflows in ERPNext.
    - Use cases: Approval processes, document lifecycle management.
- **Creating Workflows**
    - Defining states and transitions.
    - Assigning roles and permissions to transitions.
    - Example: Leave application approval workflow.
- **Advanced Workflow Features**
    - Conditional transitions (e.g., based on field values).
    - Notifications and reminders for pending tasks.
    - Debugging and troubleshooting workflows.
- **Workflow Examples**
    - Sales Order approval process.
    - Expense claim approval process.
    - Purchase Order lifecycle.

4. User Interface Customization

- **Workspace Customization**
  - Customizing navigation menus.
  - Creating custom dashboards with widgets.
  - Role-based visibility for menus and dashboards.
  - Examples: Sales manager dashboard, HR workspace.
- **Module Views**
  - Modifying list views, form views, and report views.
  - Adding filters, sorting options, and bulk actions.

5. Customization and Development

5.1 Creating an App

- Linking apps to sites.
- Commands and usage (**`bench new-app`**, **`bench install-app`**).

5.2 Managing Sites and Apps

- Managing multiple sites and apps.
- Site customization and app dependencies.

5.3 Lifecycle Hooks

- **`before_insert`**, **`validate`**, **`on_submit`**, **`on_cancel`**, **`on_trash`**.
- Practical examples of hooks in action.

5.4 Custom API Endpoints

- Writing Python scripts for custom APIs.
- Authentication and authorization.

5.5 ORM (Object Relational Mapping)

- Querying data using Frappe's ORM.
- CRUD operations (Create, Read, Update, Delete).

5.6 Writing JavaScript for Frontend

- Handling events and actions.
- Auto-name and auto-field population.

6. Advanced Customizations

6.1 Monkey Patching

- Overriding core Frappe methods safely in custom apps.
- Example: Removing the PDF button from print pages.
- Best practices for compatibility during upgrades.

6.2 Custom Web Pages and Landing Pages

- Building standalone web pages using Frappe’s framework.
- Designing attractive landing pages with custom themes.
- Dynamic content injection and routing.

6.3 Alternative Desk

- Creating a custom dashboard or workspace instead of Frappe Desk.
- Replacing Frappe Desk menus with custom navigation.
- Role-based access control for the custom desk.

6.4 Editing Core Methods

- Safely overriding core Python methods.
- Using lifecycle hooks to modify behavior.
- Testing changes and handling upgrades.

6.5 Custom Error Handling

- Throwing custom errors in APIs and Frappe Desk.
- Returning standardized error responses for REST APIs.
- Logging errors and displaying user-friendly messages.

7. Reports and Dashboards

7.1 Query Reports

- Writing SQL queries for dynamic reports.

7.2 Script Reports

- Writing Python scripts for custom reports.

7.3 Print Formats

- Customizing print layouts (PDF, HTML).
- Using Jinja templating and custom filters.
- Passing database values into print templates.

7.4 Widgets

- Adding charts, graphs, and other visual elements.

8. Integration

8.1 Payment Gateways

- Integrating payment processors (Stripe, PayPal, Razorpay).

8.2 Email Services

- Sending and receiving emails.
- Email templates and notifications.

8.3 SMS Gateways

- Integrating SMS providers (Twilio, Plivo).

8.4 Consuming External APIs

- Fetching data from third-party services.

8.5 Webhooks

- Setting up webhooks for real-time updates.

9. Deployment and Hosting

9.1 Bench Setup

- Managing production environments with Bench CLI.

9.2 Nginx Configuration

- Reverse proxy setup for high performance.

9.3 Database Backup and Restore

- Automating backups and restoring sites.

9.4 Logs

- Monitoring logs for debugging and performance tuning.

9.5 Performance Optimization

- Caching strategies.
- Database indexing and query optimization.

10. Security

10.1 Role-Based Access Control

- Permission levels and user roles.

10.2 Encryption

- Data encryption at rest and in transit.

10.3 Audit Logs

- Tracking changes and user activity.

10.4 OAuth and SSO

- Secure authentication mechanisms.

11. Advanced Topics

11.1 Background Jobs

- Running tasks asynchronously.

11.2 Cron Jobs

- Scheduling recurring tasks.

11.3 Worker Management

- Managing background workers for scalability.

11.4 Task Monitoring

- Monitoring job queues and worker performance.

11.5 Managing Multiple Sites

- Multi-tenancy and site isolation.

11.6 Frappe Fixtures

- Using fixtures for predefined data in apps.

11.7 Jinja Templating

- Using Jinja for custom print formats, email templates, and web pages.
- Writing reusable custom Jinja filters.

11.8 URL Routing

- Defining custom routes for web pages.

11.9 Dynamic Content Pagination

- Paginating large datasets in custom web apps.

11.10 Biometric Device Integration

- Integrating hardware devices with ERPNext.

12. Upgrades and Version Management

12.1 Understanding Versioning

- Major vs. minor releases.
- Reviewing release notes for changes and breaking updates.

12.2 Pre-Upgrade Checklist

- Backing up databases and files.
- Testing upgrades in a staging environment.
- Ensuring custom app compatibility.

12.3 Step-by-Step Upgrade Process

- Using **`bench update`** and **`bench migrate`**.
- Handling migration scripts and resolving conflicts.

12.4 Post-Upgrade Tasks

- Verifying functionality and fixing broken customizations.
- Cleaning up old dependencies.

12.5 Common Upgrade Issues

- Missing migrations, incompatible custom apps, worker crashes.

12.6 Examples

- Upgrading from Frappe v14 to v15.
- Debugging a failed upgrade.

13. Troubleshooting and Debugging

13.1 Common Issues and Errors

- Development errors: Syntax, validation, DocType issues.
- Deployment errors: Database migration, Nginx/Apache, worker crashes.
- Integration errors: API, webhook, email/SMS failures.

13.2 Debugging Tools and Techniques

- Frappe Desk: Tracking errors and logs.
- Browser developer tools: Debugging frontend issues.
- Logging: Writing custom logs for debugging.
- Error tracking tools: Integrating Sentry.
- Testing strategies: Writing unit tests.

13.3 Best Practices and Pitfalls

- Avoiding hardcoding, setting proper permissions, safely overriding core methods.
- Simplifying forms, streamlining workflows, ensuring scalability.
- Backing up databases, updating versions, isolating tenant data.

14. Preparing for Interviews

14.1 Core Concepts

- What is Frappe Framework? How does it differ from ERPNext?
- Explain the MVC architecture in Frappe.
- What are DocTypes, and how do they work?

14.2 Customization and Development

- How do you create a custom app in Frappe?
- What are Client Scripts and Server Scripts? Provide examples.
- How do you customize a DocType or add a new field?

14.3 Workflows and Automation

- What is a workflow in ERPNext? Provide an example.
- How do you implement role-based workflows?

14.4 Security and Permissions

- How do roles and permissions work in Frappe/ERPNext?
- What are field-level permissions, and how do you configure them?

14.5 Integration and APIs

- How do you create a custom API endpoint in Frappe?
- How do you consume external APIs in ERPNext?

14.6 Troubleshooting and Debugging

- How do you debug errors in Frappe Desk?
- What steps would you take to resolve a database migration issue?

14.7 Advanced Topics

- What is monkey patching, and when would you use it?

- How do you handle multi-tenancy in Frappe/ERPNext?

14.8 Practical Scenarios

- Design a custom module for tracking employee attendance.
- Create a workflow for approving leave applications.

14.9 Tips for Success

- Highlighting your contributions to the Frappe/ERPNext community.
- Demonstrating problem-solving skills with real-world examples.

15. Practical Projects

- **Inventory Management System** : Stock tracking, reorder levels.
- **Library Management System** : Book lending, member management.
- **Ride Booking System** : Ride scheduling, payments.
- **Custom Support Ticket System** : Issue tracking, SLA management.
- **E-Commerce Store** : Setting up an online store with ERPNext.

16. Resources and Community

16.1 Documentation

- Official Frappe/ERPNext docs.

16.2 Forums and Discussion Groups

- Engaging with the community.

16.3 GitHub Repositories

- Exploring source code and contributing.

16.4 Contributing to Frappe

- Reporting issues, submitting PRs, writing documentation.