Phase 3 – MVP Implementation

product catalog with MongoDB

GITHUB LINK:
https://github.com/Priyanka16006/Product-catalog-with-mangoDB/commits?author=Priyanka16006

NAME: Priyanka chowdry.M

COLLEGE CODE: 6128

REG NO: 612823205041

COLLEGE NAME: VARUVAN VADIVELAN INSTITUTE OF TECHNOLOGY

DEPARTMENT: B. TECH IT

1. Project Setup

Project setup is the foundational stage of the MVP Implementation. It involves preparing the development environment, installing necessary tools and dependencies, configuring the database, and organizing the project structure. For the Product Catalog with MongoDB, the setup ensures that developers can build, run, and test the application effectively.

Steps involved:

1. Environment Setup: Install Node.js, MongoDB, Git, Postman, and a code editor such as VS Code.

2. Project Initialization: Run `npm init -y` to create package.json. Install dependencies such as express, mongoose, and dotenv.
3. Database Setup: Use local MongoDB or MongoDB Atlas cloud for persistence.

4. Folder Structure: Organize files into /models,

/routes, /controllers, /config, and /tests.

5. Configurations: Create `.env` for environment variables, `.gitignore`, and initialize GitHub repo.

Example: MongoDB connection in config/db.js

```
const mongoose = require('mongoose');
mongoose.connect(process.env.MONGO_URI)
  .then(() => console.log('MongoDB Connected'))
  .catch(err => console.error(err));
```

Advantages: Organized structure, scalability, and maintainability.

Challenges: Dependency errors, environment mismatch, and initial configuration complexity.

2. Core Features Implementation

The core features represent the heart of the

Product Catalog application. In MVP, only the essential functionalities are implemented, focusing on CRUD operations and search capabilities.

Core Features:
- Add Products (Admin)
- View Products (All users)
- Update Products (Admin)
- Delete Products (Admin)
- Search and Filter by category, price, etc.
- Pagination for large datasets.

Example API for Adding Product (POST /products):

```
app.post('/products', async (req, res) => {
  const newProduct = new Product(req.body);
  await newProduct.save();
  res.status(201).send(newProduct);
});
```

Advantages: Flexible, scalable, RESTful APIs.
Challenges: Validation errors, handling large

datasets.

3. Data Storage (Local State / Database)

MongoDB is used for persistent storage in the Product Catalog. It is chosen because of its schema-less nature, scalability, and JSON-like document storage, which aligns well with product catalog requirements.

Schema Design Example:

```
----------------------------------------
const productSchema = new mongoose.Schema({
  name: String,
  description: String,
  category: String,
  price: Number,
  stock: Number,
  createdAt: { type: Date, default: Date.now
 }}},
```

Local State vs Database:

- Local State: Temporary storage on frontend (e.g., React state).

- Database: Permanent storage in MongoDB.

Advantages: Flexible schema, scalability.

Challenges: Schema design mistakes, handling large product data efficiently.

4. Testing Core Features

Testing ensures the reliability and quality of the MVP.

It verifies that CRUD operations work correctly and that the product catalog behaves as expected under different scenarios.

Types of Testing:

- Unit Testing (Jest, Mocha)
- Integration Testing (API + MongoDB)
- Functional Testing (CRUD workflows)
- Load Testing (Performance under multiple users)

Example Test with Jest:

```
---------------------------------------
test('Add new product', async () => {
  const response = await request(app)
    .post('/products')
    .send({ name: 'Laptop', price: 45000 });
  expect(response.statusCode).toBe(201);
});
```

Challenges: Test data cleanup, mocking database operations.

5. Version Control (GitHub)

Version control is essential for collaboration and maintaining history in the project. GitHub is used to track changes, manage branches, and

enable team collaboration.

Git Workflow:
- git init, git add ., git commit -m 'Initial commit'
- git branch dev, git checkout dev
- git push origin dev

Branching Strategy:
- main: Stable release branch
- dev: Development branch
- feature/*: Specific feature branches

Commit Best Practices:
- feat: add product search API
- fix: resolve update bug

Advantages: Collaboration, history tracking, and CI/CD integration.
Challenges: Merge conflicts, poor commit practices

TEAM MEMBERS:

M.G. PAVITHRA LAKSHMI

Project setup

Testing core features

M. PRIYANKA CHOWDRY

Core feature implementation

C. PAVITHRA

Data storage (local state/database)

R. SANDHIYA

Version control (GitHub)