**Functional requirements:**

1. **User Login page**: Authorization of user.
2. Passwords saved will be encrypted, so basically, **encryption and decryption of passwords** for securing sitting passwords in Database.
3. Forgot password:   
   In case, user (account already created) forgets his passwords. Now, user can **reset his password** with this functionality.
4. **Create new account**: New user profile will be created.
5. **User profile**: Setup database for user credentials, addresses, and contact information.
6. **Search bar option**: Will help users to look-up products with ease.
7. **Product categories**: Products will be categorized for convenience to find same kind of projects.
8. **Ratings database** for user to give product specific feedback and rating on a scale of

[1-5] **\*1** Bad review; **\*2** Not Satisfied; **\*3** Satisfied; **4\*** Good;

**\*5** Excellent.

1. **Filter for products (UI):** Users can be able to sort products using the below options:

* Price high – low
* Popularity based on ratings.

1. **Order cart**: Implement carting option for users to group different products and buy them at once or remove them if needed from cart.
2. **Orders placed notification!** – Email notification for user/customers who have placed their order(bought).
3. **Pagination** –   
   Pages for Website to look-up for more products instead of scrolling forever.  
   Basically, 3 - 5 products per page only.

**Non - Functional requirements:**

1. **Authentication for all GET, POST, PUT, DELETE calls.**

* Username: user
* Password: password

**With Basic Authentication**

1. **Rate Limiting and Throttling:**
2. Rate Limiting: Implement rate limiting to prevent abuse of your API and ensure fair usage.
3. Throttling: Apply throttling to control the number of requests a client can make within a given time frame.

**\*\*\*5 GET all or POST request on rating API calls**

1. **Documentation:**

API Documentation: Provide clear and up-to-date documentation for the API, including endpoints, request/response formats, and usage examples.

1. **Performance:**

Response Time: Ensure that the API responds quickly to client requests to provide a smooth user experience.

Response time less than 500 ms

1. **Authorization**: Define access controls and permissions to restrict what authenticated users can do within the API.
2. **Error Handling**: Implement proper error handling and logging to aid in debugging and troubleshooting.
3. **Test Automation**: Implement automated testing to catch regressions and issues early in the development process.

**Project plan**

1. UI Plan front-end React.js  
2. database schema -- mongoDB

3. Api class -> middle Django python

4. User story – Jira

**Functional requirements:**

1. Login page: Authorization
2. Forgot password
3. Create new account
4. User profile
5. Encryption - fdecryption
6. Setup database for credentials
7. Search bar
8. Product categories/ catalog
9. Filter for products

Price high – low

Popularity based on ratings

1. User-ratings database
2. Carts
3. Orders
4. Pagination

**Non - Functional requirements:**

1. **Authentication for all GET, POST, PUT, DELETE calls.**

Username: user

Password: password

**With Basic Authentication**

1. **Rate Limiting and Throttling:**
2. Rate Limiting: Implement rate limiting to prevent abuse of your API and ensure fair usage.
3. Throttling: Apply throttling to control the number of requests a client can make within a given time frame.

5 api calls per min

**\*\*\*5 GET all or POST request on rating api calls**

1. **Documentation:**

API Documentation: Provide clear and up-to-date documentation for the API, including endpoints, request/response formats, and usage examples.

1. **Performance:**

Response Time: Ensure that the API responds quickly to client requests to provide a smooth user experience.

Response time less than 500 ms

1. Authorization: Define access controls and permissions to restrict what authenticated users can do within the API.
2. Error Handling: Implement proper error handling and logging to aid in debugging and troubleshooting.

Test Automation: Implement automated testing to catch regressions and issues early in the development process.