

**Full Stack .NET Developer**

Capstone Project Problem Statement





**Capstone Project: E-Healthcare**

**Problem statement:**

Create a dynamic and responsive .Net+Angular e-healthcare web application to allow users to purchase medicines for different categories.

**Background of the problem statement**

ABC Healthcare is a Mumbai-based pharma company which sells medicines throughout India as per the demand it receives from multiple hospitals and clinics. It has been operating in India for the last 15 years.

However, in recent years since all industries are coming online to boost their sales and cover larger market space, ABC Healthcare missed the chance and as a result their sales have been dropping for the last 2 years. To overcome this and bring sales back on track they have decided to bring their service online to allow their customers to purchase medicines through their web application.

You are hired as one of the Full Stack .Net developers and have been asked to develop the web application. The management team has provided you the requirements and their business model so that you can easily arrange different components of the application.



**Features of the application:**

1. User can register

2. User can log in

3. User can log in as admin

4. User can add/update medicines to portal

5. User can browse through medicines

6. User can search medicines

7. User can sort medicines based on name and category

8. User can manage the cart

9. User can place order

**Recommended technologies:**

1. Database management: SQLServer

2. Back-end logic: ASP.NET Web API

3. Front-end development: ReactJS,Angular, HTML/CSS, and Bootstrap

4. Testing technologies: NUnit/XUnit

5. DevOps and production technologies: Git, GitHub, Jenkins, Docker, and Azure



**Project development guidelines:**

1. The project will be delivered within four sprints with every sprint delivering a  minimal viable product.
2. It is mandatory to perform proper sprint planning with user stories to  develop all the components of the project.
3. The learner can use any technology from the above-mentioned technologies  for different layers of the project.
4. The web application should be responsive and should fetch or send data  dynamically without hardcoded values.
5. The learner must maintain the version of the application over GitHub and  every new change should be sent to the repository.
6. The learner must implement a CI/CD pipeline using Jenkins.
7. The learner should also deploy and host the application on an AWS Azure.
8. The learner should make a rich front-end of the application, which is user friendly and easy for the user to navigate through the application.
9. There will be two portals in the application, namely the admin and user  portal. More information on this is on the next page.
10. Application should be responsive and user friendly on mobile devices.



**Admin Portal:**

Admin is responsible for managing content in the portal. Admin will handle adding and updating data in the portal.

Admin should be able to perform below functionalities:

1. Login in to the system
2. View list of medicines in the table format
3. View list of medicine categories in table format
4. Create a new category with name
5. Add new medicines in categories with name, price, image, seller, and descriptions
6. Update existing medicines in categories with name, price, and image.
7. Disable medicines from categories which are not available
8. View list of users in table format

**User Portal:**

From the user’s perspective, the application should allow users to browse through different medicines, add them to their cart, and purchase them. However, the user should be logged in to make any purchase through the portal.

Users should be able to perform below functionalities:

1. Browse through different medicines available in the portal
2. Register in the application if not done yet
3. Login into the portal before making any purchase
4. Add medicines which you want to buy in the cart
5. Place order for medicines in the cart
6. Get Order summary when order is placed