Agenda

1. Operational Concept Overview
2. Live Demo
3. Testing and test cases
4. Quality Focal Point
5. Transition Plan Summary
6. Questions and feedback
OPERATIONAL CONCEPT OVERVIEW
# Shared Vision

## The Program Model

| Assumptions |
|---|---|---|---|
| **Stakeholders** | **Initiatives** | **Value Propositions** | **Beneficiaries** |
| Client | Weekly meeting with client to make sure we are on the right path | Adding the features as requested would help increase user engagement | The users using the blog |
| Designer | Sync up with designer so that the UI is as expected by the client/UX experts | The comment sanitization will help reduce spam and abusive comments | The client since it will save their development team’s resources and time. |
| Development Team | Weekly team meetings to discuss progress | The search feature will help users to find relevant blogs. | |
| Blog Users | | | |
| Expert Dieticians | | | |
System Capabilities and Description

- Targets “diet conscious” and “fitness enthusiastic” people.
- Allow users reading the blog to comment and like a blog post.
- Admin panel helps maintainers reduce manual effort.
- A more customized blog publishing platform would help us differentiate from our competitors.
- An automated CICD system allows developers to deploy code automatically to AWS.
Expected Benefits

- **Users** Blog visitors/End Users can comment/like a blog post and even search them.

- **Maintainers** Administrators can use a customized admin panel to manage already published blogs or even publish a new blog.

- Comment sanitization system helps report vulgar comments in real time.

- **AWS Auto Deployment** feature helps developers to quickly test and deploy code.
Benefit Chain

Assumptions:
- Stable network and database systems
- Decent participation by users in surveys and testing
- Blogs are published in English language only

Figure 1: Benefits Chain Diagram of Meta Nutrition Blog.
System Boundaries & Environment

- Comment sanitization System
- Admin panel for blog
  publish/edit/delete
- Automated deployment system for CI/CD
- Like/Comment/Share feature for blog posts

Support Infrastructure:
- NextJS
- NodeJS
- React
- MongoDB
- AWS

- Expert Dieticians
- End Users
- Client
- System Admin/Developers
- Comment Sanitization System
- CICD Automated Deployment
- Admin Panel
- Internet
System Constraints

- **CO-1: AWS as deployment environment**: The new system should be able to run and be compatible with AWS and there should be no cost involved with AWS free tier currently.

- **CO-2: NextJS as a framework**: NextJS and NodeJS will be used during development of the UI and backend modules.

- **CO-3: MongoDB as a database**: MongoDB should be used as a database to store information and should be hosted on AWS.

- **CO-4: GitHub as source repository**: GitHub should be used where all the code is to be stored.
TEST CASES
Test case scenarios

Admin Side

- Admin login
- Post addition
- Post deletion

User Side (Blog visitor)

- Posting comments on a post
- Liking a post
- Searching for posts with keywords

Development

- Auto Deployment Feature
Testing Strategy

● Manual, black box testing for all the core capabilities

● Test Process
  ○ Identifying all the core capabilities to be tested
  ○ Creating test cases to exhaustively test all possible scenarios for each core capability
  ○ Executing test cases
  ○ Finding and reporting bugs
  ○ Fixing bugs
  ○ Repeat
Admin Login

Goal
● Cover all scenarios where admin login can fail
● Ensure that admin login is successful when the credentials are correct

Procedure
● Fill up the input form as per the test case specification and submit it
● Check if the resulting behaviour is as expected in order to fulfil passing criteria
<table>
<thead>
<tr>
<th>Test Case Number</th>
<th>Test Item</th>
<th>Procedure</th>
<th>Pass Criteria</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-01-01</td>
<td>Email does not exist while login</td>
<td>Enter an email that is not present in the database and submit form</td>
<td>Displaying error stating “Email does not exist”</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-01-02</td>
<td>Incorrect password</td>
<td>Enter a password that is incorrect and submit form</td>
<td>Displaying error stating “Password is incorrect”</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-01-03</td>
<td>Successful login</td>
<td>Enter correct email password combination in login form</td>
<td>Admin successfully logged in and routed to dashboard</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Post Addition

Goal

- Cover all scenarios where post addition can fail
- Ensure that when a post is successfully added:
  - It visible in the first row on admin dashboard
  - It is visible on user side

Procedure

- Fill up the input form as per the test case specification and submit it
- Check if the resulting behaviour is as expected in order to fulfil passing criteria
## Test Cases for Post Addition

<table>
<thead>
<tr>
<th>Test Case Number</th>
<th>Test Item</th>
<th>Procedure</th>
<th>Pass Criteria</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-02-01</td>
<td>Post title field left empty</td>
<td>Title field left empty while submitting form for adding new post</td>
<td>Displaying error below title field stating “Title is required”</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-02-02</td>
<td>Post body field left empty</td>
<td>Body field left empty while submitting form for adding new post</td>
<td>Displaying error below body field stating “Body is required”</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-02-03</td>
<td>Post cover image not added</td>
<td>Cover image not added while creating a new post</td>
<td>Displaying error below cover image field stating “Cover image is required”</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-02-04</td>
<td>Post added successfully</td>
<td>None of the required fields left empty while submitting form for adding new post</td>
<td>Route admin to the dashboard where the new post is visible in the first row of the “posts” table</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Post Deletion

Goal
● Cover all scenarios where post deletion can fail
● Ensure that when a post is successfully deleted:
  ○ It is not visible on admin dashboard
  ○ It is not visible on user side
  ○ It is deleted from database

Procedure
● Press the delete button next to a post on the admin dashboard
● Click confirm on the pop up to confirm deletion
## Test Cases for Post Deletion

<table>
<thead>
<tr>
<th>Test Case Number</th>
<th>Test Item</th>
<th>Procedure</th>
<th>Pass Criteria</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-03-01</td>
<td>Modal pop up on pressing delete button</td>
<td>Pressing the delete button next to a post on the admin dashboard</td>
<td>A modal pop up asking the admin to confirm deletion should be displayed</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-03-02</td>
<td>Post not deleted if cancel button pressed on popup</td>
<td>Pressing the delete button next to a post on the admin dashboard and then pressing cancel button in the pop up</td>
<td>Pop up disappears and post not deleted</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-03-03</td>
<td>Post deleted if delete button pressed on popup</td>
<td>Pressing the delete button next to a post on the admin dashboard and then pressing delete button in the pop up</td>
<td>Pop up disappears and post is deleted</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Comment Feature

Goal

- Cover all scenarios where comment feature can fail
- Ensure that when a comment is successfully posted:
  - It is visible in the comment section with the commenter's name, comment text and date.
- Ensure that when a comment is not posted:
  - Appropriate helper modal is displayed guiding the user to the correct way of posting comments.
Comment Feature

Procedure

● Fill up the comment input boxes with name and comment text
● Check if the comment is posted successfully if the comment posting criteria are met: no vulgar words, non-empty comment text
● Check if the appropriate helper modal is displayed if the comment posting criteria are not met and comment is not posted
## Test Cases for Comment Feature

<table>
<thead>
<tr>
<th>Test Case Number</th>
<th>Test Item</th>
<th>Procedure</th>
<th>Pass Criteria</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-04-01</td>
<td>Check if modal is displayed when comment is empty</td>
<td>Submit the comment with empty comment text field</td>
<td>Display helper modal, asking user to enter their comment</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-04-02</td>
<td>Check if modal is displayed when comment contains vulgar language</td>
<td>Enter vulgar language in either the name or the comment field or both</td>
<td>Display disclaimer modal, asking user to not post vulgar comments</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-04-03</td>
<td>Check if comment is posted on comment section</td>
<td>Enter comment that passes all success criteria: no vulgar language, comment text not empty</td>
<td>Comment is posted in comment section</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Like Feature

Goal
- Cover all scenarios where Like feature can fail
- Ensure that when a post is liked:
  - The empty heart icon changes to filled heart icon and like count is incremented by 1
- Ensure that when a post is unliked:
  - The filled heart icon changes to empty heart icon and like count is decremented by 1

Procedure
- Like/unlike a post and check whether heart icon and like count are updated
## Test Cases for Like Feature

<table>
<thead>
<tr>
<th>Test Case Number</th>
<th>Test Item</th>
<th>Procedure</th>
<th>Pass Criteria</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-05-01</td>
<td>Check if like is triggered on click of empty heart icon</td>
<td>Like the post by clicking on the empty heart icon</td>
<td>Filled heart icon is displayed in place</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-05-02</td>
<td>Check if unlike is triggered on click of filled heart icon</td>
<td>Unlike the post by clicking on the filled heart icon</td>
<td>Empty heart icon is displayed in place</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-05-03</td>
<td>Check if like counted is updated</td>
<td>Check if like count is increased by one when user likes a post, decreased by one when they unlike a post</td>
<td>Like count increased by one on like and decreased by one on unlike</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Search Feature

Goal

- Cover all scenarios where Search feature can fail
- Ensure auto suggest drop down is displayed, matches are displayed by category when selected, “No posts to display” is displayed when there are no matches

Procedure

- Search for a keyword, select a category
## Test Cases for Search Feature

<table>
<thead>
<tr>
<th>Test Case Number</th>
<th>Test Item</th>
<th>Procedure</th>
<th>Pass Criteria</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-06-01</td>
<td>Check if auto suggest drop down appears when user begins typing (searching)</td>
<td>Enter keywords likely to contain a match in the search box</td>
<td>Autosuggest dropdown appears according to the entered keywords</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-06-02</td>
<td>Check if “No posts to display” is displayed on no matches</td>
<td>Enter random keywords which will not have any matches</td>
<td>“No posts to display” message is shown</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-06-03</td>
<td>Check if matches are displayed according to category when a category is selected (Health, Lifestyle, Vegan)</td>
<td>Search for a keyword and select a category. Check if posts with the keyword and selected category are displayed</td>
<td>Posts belonging to selected category and containing the entered keyword are displayed</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Auto Deployment Feature

Goal

- Show the stages of Auto Deployment and where it can fail
- Ensure that bug free code is automatically deployed on the live site when it is pushed on the production branch on Github

Procedure

- Push code on the production branch of Github
# Test Cases for Auto Deployment Feature

<table>
<thead>
<tr>
<th>Test Case Number</th>
<th>Test Item</th>
<th>Procedure</th>
<th>Pass Criteria</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-07-01</td>
<td>Check if deployment errors are visible in AWS console</td>
<td>Push buggy code to the production branch on Github that will cause deployment errors in AWS console</td>
<td>Deployment Errors are visible in the AWS console and the code is not pushed to the live site</td>
<td>Pass</td>
</tr>
<tr>
<td>TC-07-02</td>
<td>Check if code changes are reflected on live site</td>
<td>Push bug free code to the production branch on Github</td>
<td>The code and its corresponding changes are visible on the live site (The changes are deployed)</td>
<td>Pass</td>
</tr>
</tbody>
</table>
Quality Focal Point
QFP

1. Traceability Matrix
2. Technical Debt
## Traceability Matrix

<table>
<thead>
<tr>
<th>OCD</th>
<th>Requirement</th>
<th>Use Case</th>
<th>Test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Like/Comment</td>
<td>WC-279</td>
<td>UC-1</td>
<td>TC-05-01, TC-05-02, TC-05-03</td>
</tr>
<tr>
<td>Admin Panel</td>
<td>WC-278</td>
<td>UC-2</td>
<td>TC-01-01, TC-01-02, TC-01-03, TC-02, TC-03</td>
</tr>
<tr>
<td>Comment sanitization</td>
<td>WC-277</td>
<td>UC-3</td>
<td>TC-04-01, TC-04-01, TC-04-01</td>
</tr>
<tr>
<td>Automated Deployment</td>
<td>WC-276</td>
<td>UC-4</td>
<td>TC-07-01, TC-07-02</td>
</tr>
</tbody>
</table>
Technical Debt

- Lack of documentation
- Migrating MongoDB to AWS
- Lack of unit tests
- Website responsiveness
Transition Plan Summary
Transition Plan Overview

- **Hardware Transition** - Not Applicable to our project
- **Software Transition** -
  - Transfer Github repository ownership to Meta Nutrition Organization.
  - Daily Office hours till 12/3/2020 to solve any issues faced by the client.
  - Handoff of final documents and user manual
- **Staff Preparation** -
  - Conducting weekly meeting / office hours since last 4 weeks to make client familiar with the project
  - Capability walk through for the admin panel
  - Training admin on using the Customized Editor
## Stakeholder Responsibilities

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Supporting roles</th>
<th>Supporting Skills</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>System and database administration</td>
<td>Technical knowledge such as NodeJS, React, AWS, MongoDB. Leadership skills</td>
</tr>
<tr>
<td>Developers</td>
<td>Software maintenance and development, database maintenance, operational support</td>
<td>Technical knowledge such as NodeJS, React, AWS, MongoDB</td>
</tr>
<tr>
<td>Blog Visitors</td>
<td>Software users, feedback providers, blog visitors, commenting and liking blog posts</td>
<td>Basic browser navigation knowledge</td>
</tr>
<tr>
<td>Expert Dieticians</td>
<td>Blog content providers, content writers</td>
<td>Content writing skills related to healthy food eating habits, fitness and healthy lifestyle.</td>
</tr>
</tbody>
</table>
## Milestone Plan

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Team Members</th>
<th>Deadline Date</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Manual</td>
<td>Aman Mahajan, Elena Ramezani</td>
<td>11/13/2020</td>
<td>Completed</td>
</tr>
<tr>
<td>Unit Test Cases</td>
<td>Aman Mahajan, Soham Chitalia</td>
<td>11/16/2020</td>
<td>Completed</td>
</tr>
<tr>
<td>Training Session 1 - Editor Functionalities</td>
<td>Ayush Jain, Akshay Jain</td>
<td>11/16/2020</td>
<td>Completed</td>
</tr>
<tr>
<td>Training Session 2 - Code Walk-Through</td>
<td>Adit Dharia, Elena Ramezani, Soham Chitalia</td>
<td>11/23/2020</td>
<td>In Progress</td>
</tr>
<tr>
<td>Training Session 3 - Admin Dashboard</td>
<td>Adit Dharia, Ayush Jain</td>
<td>11/23/2020</td>
<td>In Progress</td>
</tr>
<tr>
<td>Transfer Github Repository</td>
<td>Arnav Jalui, Akshay Jain</td>
<td>11/27/2020</td>
<td>In Progress</td>
</tr>
<tr>
<td>Final Documents Handoff to Client</td>
<td>All Team Members</td>
<td>11/27/2020</td>
<td>In Progress</td>
</tr>
</tbody>
</table>
User Manual

- **How to use the Admin Panel**
  - Steps to navigate through the Admin dashboard
  - Steps to manage all the blog articles (add, edit, delete)
  - Screenshots provided for all steps

- **How to use the Meta Nutrition Blog website (for visitors)**
  - Steps to navigate through the website
  - Steps to use all the core capabilities in the website

- **How to use GitHub Auto-deployment using AWS CodeDeploy**
  - Steps to access GitHub repository and pushing changes to the repository
  - Steps to access AWS CodeDeploy portal
  - Screenshots provided with detailed information on every step
Technical Manual

● Documentation
  ○ Steps for Clients/developers to run the application locally in development environment
  ○ Steps for installation of various dependencies. (React/nextJS, Node, etc)
  ○ FAQ regarding troubleshooting
  ○ Steps for hosting of the Meta Nutrition Blog on AWS
  ○ Steps to access AWS account
  ○ Steps for using AWS CodeDeploy for GitHub auto-deployment
  ○ Steps to use MongoDB to manage blog data

● Credentials to access services
  ○ Includes credentials for services like AWS, MongoDB, AWS CodeDeploy, GitHub.
Training Sessions

- **Code walk-through**
  - Explain the client about the code structure and various files

- **How to use Admin Dashboard**
  - Explain the client about all core capabilities in detail
  - Explain how to use the customized editor

- **How to use MongoDB**
  - Explain the client about how to access MongoDB
  - Explain manipulating data in MongoDB (add, edit, delete data)
  - Explain use of MongoDB Compass for data manipulation
  - Explain use of indexes in MongoDB and how you can create one

- **Troubleshooting sessions**
  - We will be conducting weekly meeting/office hours in the last 4 weeks to make client familiar with the project
Thank You