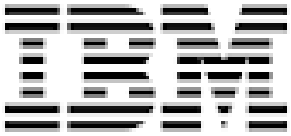


COLLEGECODE:9126

COLLEGE NAME: SRI BHARATHI ENGINEERING COLLEGE FOR WOMEN

DEPARTMENT:BE-CSE

STUDENT-NM-ID:au912623104007

Au912623104011

Au912623104013

ROLL.NO :912623104007

912623104011

912623104013

DATE :

COMPLETEDTHEPROJECTNAMEAS:IBM-NJ-ONLINEQUIZAPPLICATION PHASE-TECHNOLOGY PROJECT NAME:NODE JS

SUBMITTEDBY,

V.BRINTHA

R.DHARSHINI

R.GAYATHIRI

MOBILENO:

9787026737

6382996269

784507G586

Phase3: MVP ImplementationCDeadline-Weeks

Project setup

1. DefineProjectScopeandObjectives

- **Goal Clarity**: Understandanddocumenttheoverall goalofthe project. What problem are you solving or what value are you providing?
- **Deliverables**:Listthemajoroutputsandresultsexpectedfromtheproject.
- **SuccessMetrics**:Definehowsuccesswillbemeasured(e.g.,performance metrics, user satisfaction, financial goals).

2. IdentifyStakeholdersand Team

- **Stakeholders**: Determinewho will beimpacted by orhavean interestintheproject, such as clients, users, and other departments.
- **RolessResponsibilities**:Assign clearrolestoyourteammembers.Ifyou're workinginateam,clarifywho'sdoingwhat.

3. SetUpVersionControlandProjectManagementTools

- **Version Control (Git, GitHub, GitLab, Bitbucket, etc.)**: Even if you're working alone, version control is essential for tracking changes, collaborating, and maintaining the project history.
- **ProjectManagement(Jira,Trello,Asana,Monday.com,etc.)**:Organizetasks,setdeadlines, assignteammembers,andkeeptrackofprogress.

CoreFeaturesImplementation

1. UserAuthenticationsAuthorization

Features:

- UserregistrationClogin(OAuth2.0,JWT)
- Role-basedaccess(Admin,Instructor,Student)

Implementation:

- **Backend:**SpringSecuritywithJWTforstatelessauth
- **Frontend:**TokenstoredinHttpOnlycookieorlocalStorage

2. QuizManagement(Admin/Instructor)

Features:

- Create,update,deletequizzes
- Addquestions(MCQ,True/False,ShortAnswer,CodeSnippets)

Implementation:

- **Backend:**RESTAPIswithCRUDendpointsforquizzes/questions
- **Database:**IBMDb2/PostgreSQL onIBMCloud

DataStorage(LocalState/Database)

1. Local State (Client-Side Storage) - For Temporary, Session-Specific Data

- **TemporaryUser Progress (Current Session):**Local state (like browser **sessionstorage** or an application state management tool) is best for tracking a user's progress **duringthe active, single quiz session**.

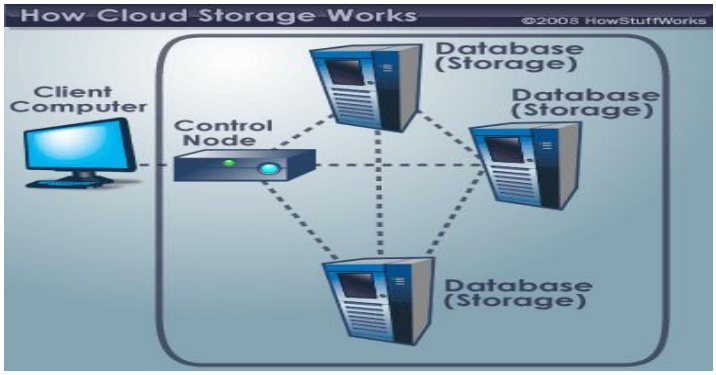
Example: Storingthe user's selected answer for the current question beforethey click "Next.

- **ImprovedResponsiveness/Performance:**Storingnon-critical,frequentlyaccessedstatelocallyreducestheneedforconstantnetworkcallstothe server,resultinginafaster,smootheruserexperience.

2. Database(PersistentServer-SideStorage)-EssentialforCoreQuiz Functionality

DataIntegrityandPersistence: Thedatabaseismandatoryforstoring **critical,persistent data**, such as:

- **Quiz Questions and Answers (Correct ones):** Must be secure on the server to prevent cheating.
- **User Registration/Authentication Data (IBMid, User Profile):** Required for identifying the user.
- **Security:** Storing correct answers and final results server-side (in a database) is essential for security and preventing client-side tampering/cheating.



TestingCoreFeatures

1. UserAuthenticationandAccess

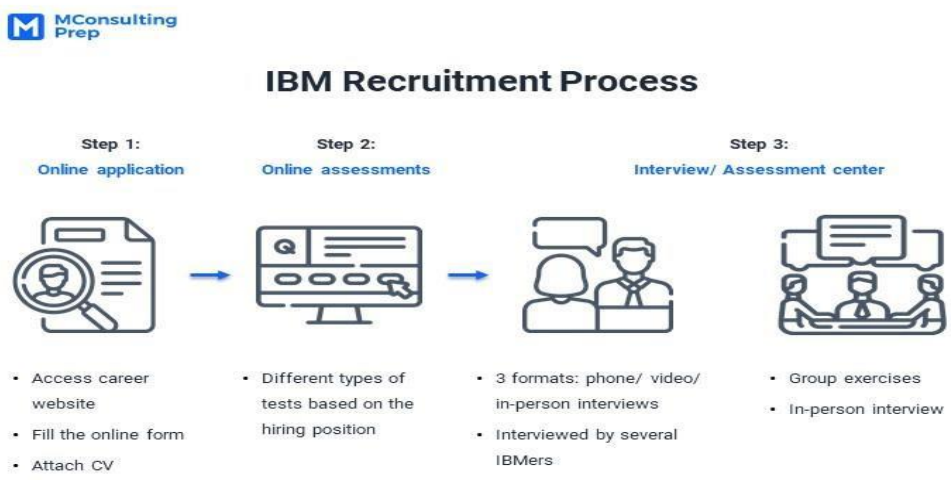
- **Login/Logout:** Test with valid/invalid credentials, session management, and proper logoutfunctionality.
- **User Roles/Permissions (if applicable):** Ensure a "student/candidate" can only take a quiz, and an "administrator/instructor" can create/edit quizzes and view results.

2. QuizDeliveryandFlow

- **Start/Submit Quiz:** Verify the quiz starts correctly (e.g., timer starts, questions load) and submits correctly, saving all answers.

3. ScoringandResults

- **Answer Key Mapping:** Test that the system correctly maps submittedanswersto the correct answers defined by the author.
- **ScoringLogic:**
- Testforscenarioslikepartialcredit,negativemarking(ifconfigured),and unattempted questions.



VersionControl(Githup)

CoreGit/VersionControlPractices

- **Branching Strategy:** Usea structured branchingmodel (like Gitflow ora simplified feature-branch workflow) to isolate development.
- **`main(ormaster)`:**Keepthisbranchstableanddeployable,representingthe production code.
- **AtomicandDescriptiveCommits:**Make**small,logical,andfrequentcommits** that represent a single, complete unit of work (e.g., "Implement question loading logic,"not"Massiveupdate").

`gitignoreFile`: Usethistoexcludefilesthatshouldn'tbetrackedbyGit,suchas

Dependencyfolders(e.g.,`node_modules/`,`vendor/`)

UsingGitHubFeaturesforCollaborationandDevelopment

- **Remote Repository:** UseGitHubtohost yourremoterepository, providingacentral backup and a platform for collaboration.
- **Issue Tracking:**Use **GitHub Issues**totrackbugs, plannewfeatures(e.g., "Implementa'resetquiz'button"),andmanagetasks.Assignissuestospecific teammembersandlinkthemtorelevantcommitsorpullrequests.

GitHubActions(CI/CD): AutomatepartsofyourworkflowusingGitHubActions.Foran online quiz, this could include:

- **Continuous Deployment (CD):** Automatically build and deploy the application to a staging or production server once a branch (like `main`) is updated.