**ENTREPRENEURIAL CODE CRAFTERS**

**Sprint 2: Gather requirements**

**Considerations: Are any technology investigations needed?**

**Database Technology:**

* Research and analyze database solutions in Visual Studio, including SQL Server and Azure SQL Database. Consider data volume, flexibility, real-time coordination, and interoperability with existing systems.

**Real-time updates for investigation:**

* Investigate C# tools or frameworks that provide real-time communication and modifications among the client-side UI and the server-side database. Evaluate the performance and sensitivity of real-time capabilities under different loads.  
  Evaluate available reporting solutions in Visual Studio, including Microsoft Reporting.
* Services and third-party libraries

**Integration into ERP systems:**

* Investigate the APIs or SDKs systems to ensure easy connectivity with the system used for inventory management. Check the authentication procedures, data formats, and interoperability with C# for data interchange and synchronization.

**Security and Authentication:**

* Investigation: Use C# in Visual Studio to implement authentication and permission protocols. Consider options such as ASP.NET Identification for secure authentication of users and role-based control of access.

**UI/UX Development:**

* Investigation: Use Visual Studio's capabilities to create responsive UI/UX interfaces in C#, ASP.NET MVC, or ASP.NET Core. Consider using frontend frameworks like as Bootstrap or React to improve user experience and performance.

**Testing, debugging, and investigation:**

* Set up frameworks for unit testing in Visual Studio to test C# code automatically. Debugging instruments and profilers can help you detect and fix bottlenecks in performance, memory leaks, and runtime issues.

**Deployment and DevOps: Investigation:**

* Set up CI/CD pipelines in Visual Studio Team Operations (Azure DevOps) for ongoing integration and C# code deployment. Investigate options for control of versions, code reviews, automation builds, and delivery to environments for production.

# Team Availability: Lilitha Njeje - Database Administer (Available full-time)

# Avarn Sewlal - UX/UI Designer (Available full-time)

# Allison Gopal - UX/UI Designer and Developer (Available full-time)

# Hayley Hananiah Chetty – Technical Writer (Available full-time)

# Sibusisiwe Sbahle Kunene – Software Architect(Available fulltime)

# Malibongwe Nzuzo Ndlovu – Project Manager (Available full-time)

# Requirements responsibility

# Acceptance Criteria:

* Collaborate with product owners and stakeholders to acquire precise application requirements.
* Prioritize demands based on their commercial value, influence on users, and their technical viability.
* Use technologies such Visual Studio TFS or Azure Development Operations to create clear and structured requirements documentation.
* Ensure requirements are clear, explicit, and related to user needs.
* Validate needs with the client to ensure comprehension and alignment.
* Regularly update the backlog to reflect comments, changes in priorities, and new insights from stakeholders.
* Inform the development team about any requirements-related dependencies or constraints.

**Tasks:**

* Schedule meetings with product managers and stakeholders to review needs.
* Implement Visual Studio TFS or Azure Development Operations to manage the backlog.
* Regular backlog grooming meetings help refine and prioritize demands.
* Document thorough user stories, acceptance standards, and technical specifications for needs.
* Obtain clarity from the customer or stakeholders for any ambiguous needs.
* Regularly update the backlog with fresh data, revisions, or additions to reflect project progress.
* Effectively communicate with the software development team to ensure everyone knows and aligns with prioritized requirements.

**Create first user journey – UX responsibility**

**Acceptance Criteria:**

* Work along with the UX designer to draft a user journey map that describes how displays flow and how users interact with them.
* Based on business logic and customer requirements, define user roles and permissions.
* Verify that essential features, functionalities, and navigation routes are included in the user experience.
* Confirm the user journey with stakeholders and make any necessary adjustments in light of their input.
* Use software like Adobe XD, Figma, Sketch, or other visual documentation tools to create wireframes or flowcharts that depict the user path.
* Effectively convey the user journey to the development team so that it can direct UI design.

**Tasks:**

* Arrange a launch meeting to go over features, user roles, and preliminary thoughts with the UX designer.
* Work together to create wireframes or mock-ups that illustrate the screen interactions and user journey.
* Specify the user roles, degrees of access, and permissions needed for each section of the program.
* The screens, actions, and transitions should be mapped out according to user stories and business logic.
* Show stakeholders the initial user journey so they can offer suggestions and edits.
* Provide the development team with a clear and easily readable structure in which to record the completed user journey.
* Let the UX designer know about any technological issues or limitations pertaining to the user journey.

**Frontend responsibility**

**Acceptance Criteria:**

* Examine various frontend frameworks in light of the needs of the project, their scalability, developer experience, and community support.
* Select a frontend framework that fits the architecture, UI/UX requirements, and performance factors of the project.
* For clean, maintainable, and effective code, adhere to industry best practices and coding standards.
* Use responsive design concepts to make sure the program functions properly on a range of screens and devices.
* Manage code updates and modifications by working with the development team and utilizing version control systems.
* Perform testing, debugging, and code reviews to find and fix problems early in the development process.
* For future reference and maintainability, document the code structure, components, and dependencies.
* To keep your coding abilities up to speed, keep up with frontend trends, emerging technologies, and best practices.

**Tasks:**

* Based on the limits and requirements of the project, investigate and contrast frontend frameworks.
* Using the selected frontend framework, create a prototype or proof of concept to show functionality and viability.
* Based on industry best practices, establish coding standards and guidelines for HTML, CSS, and JavaScript.
* Use CSS media queries, flexible layouts, and viewport meta tags to implement responsive design.
* Establish a Git repository for version control and use branching and merging techniques to work together with the development team.
* Review code with colleagues to make sure it follows standards and is of high quality.
* Test front-end components and interactions through unit, integration, and end-to-end testing.
* In a README or documentation file, describe the code structure, component hierarchy, API integrations, and third-party dependencies.

**DevOps responsibility**

**Acceptance Criteria:**

* Get a thorough grasp of all the components of GitHub Actions, such as jobs, triggers, workflows, and actions.
* Using project requirements, code quality standards, and testing requirements as a basis, specify the phases and procedures of the CI/CD pipeline.
* Set up GitHub Actions workflows to automate the activities of developing, deploying, testing, and validating code.
* Include code quality checks in the pipeline, such as unit tests, integration tests, static code analysis, and linting.
* To enable prompt response and resolution, set up notifications and alerts for pipeline status, failures, and deployment outcomes.
* Verify that the pipeline is flexible, scalable, and adaptable to various environments, deployment targets, and branches.
* Work together to verify and enhance the CI/CD pipeline for dependability and efficiency with developers, testers, and stakeholders.

**Tasks:**

* Acquire knowledge about GitHub Actions via tutorials, documentation, and practical experience.
* Determine which tests and code quality checks should be included in the CI/CD pipeline and rank them accordingly.
* Within the GitHub repository, specify task parameters and workflows in YAML language.
* Set up triggers to start pipeline runs, such as pull requests, push events, or scheduled jobs.
* Install and use custom scripts or third-party actions for particular tasks, like deployment, code formatting, and dependency management.
* Use optimizations and caching techniques to expedite pipeline execution and lower resource usage.
* For sensitive data and configuration options, safely set up secrets and environment variables within GitHub Actions.
* Keep an eye on metrics, logs, and pipeline performance to spot problems, bottlenecks, and potential improvement areas.

**Backend responsibility**

**Acceptance Criteria:**

* Determine the SDK versions, libraries, and frameworks needed for backend development based on the specifications of the project and the technological stack.
* Keep track of the suggested versions and dependencies in a single place, such the project documentation or a README file.
* Make certain that everyone in the team has access to the suggested versions and that all local development environments have the same configurations.
* Use tools like package managers (npm, NuGet) or dependency management files (package.json,.csproj) to set up version control for project dependencies.
* Check that the libraries are integrated and compatible with the IDE, development tools, and programming language of choice.
* Inform the team of any upgrades or modifications to the library versions and help in updating local environments as necessary.

**Tasks:**

* Examine the project specifications, architecture, and technological stack to ascertain the required SDK versions and backend libraries.
* Verify library selections and versions by speaking with project leads, architects, or senior developers.
* Provide installation guidelines, compatibility notes, and a list of suggested library versions and dependencies.
* Use Git or another version control system to set up version control for backend code and dependencies.
* Set up development environments to utilize the suggested SDKs and library versions.
* Before committing code changes, verify that libraries are compatible and working in the local development environment.
* Help team members with debugging environment setup, updating libraries, and resolving dependency issues.
* During the course of the project, keep up with documentation and upgrade library versions as necessary.