Development Plan

ShareWize: Expense Sharing Web App

Group 24

Arun Rathaur

Ayaz Lockhat

Mubtasim Rahman

Veeral Patel

October 27, 2023

ShareWize

SHARING MADE SIMPLE

Revision History

Version	Date	Change	Revision
0	October 27, 2023	-	-

1. Team Meeting and Communication Plan

- **Google Drive:** Google Drive is an excellent tool for collaborative document sharing and storage. Here's how we plan to use it:
    - **Folder Structure:** A projects folder on Google Drive, and within it, sub-folders for different aspects of the project. Such as our project plan, design documents, reports etc.
    - **Document Sharing:** These folders will be shared with the team so everyone has access to the documents within it. Google Drive provides an easy and seamless way for this.
    - **Version Control:** We will also utilize Google Drive's version history to track changes in documents.
- **GitLab:** GitLab is an excellent tool for version control and collaboration. Here's how we plan to use it:
    - **Repository Setup:** We will create a Gitlab repository for the project. We will create separate branches for different features or tasks.
    - **Code Review:** The team can use code review features that ensure code changes are checked and approved before merging into the main branch.

- **Discord:** This will serve as a real-time communications and meetings platform where the team will communicate the project in a call and in chat.
    - **Dedicated Channels:** For better organization we will have different channels for different purposes. Such as “Code Discussion”, “General”, “Meeting chat”, etc.
    - **Schedule Meetings:** We will communicate meeting times using the Discord voice channel. This will allow us to communicate effectively.
    - **Screen Sharing:** We will utilize Discord’s screen sharing feature during meetings for presentations, code reviews, or any other reason.
- **TeamGantt:** We have already started to track the timeline of our project through this project management platform with the start of the Gantt chart which can be found at the bottom of our document. We plan on using it for:
    - **Assigning tasks**: We have already started doing so and will continue to use it as the project develops.
    - **Creating Baselines**: In the case of shifting project timelines.
    - **Using Notes:** Leave them on functional and nonfunctional requirement tasks.

2. Team Member Roles

Name	**Role**
Arun Rathaur	Developer (Data and Security)
Ayaz Lockhat	Developer (Front-End and Database)
Mubtasim Rahman	Project Manager and Developer (Front-End and Back-End)

Veeral Patel	Developer (Back-End and QA Testing)

Requirement (F/NF)	**Contributors**
Authentication - F	Mubtasim, Veeral,Arun
Expense Tracking - F	Ayaz, Mubtasim, Veeral
Expense Sharing - F	Ayaz, Mubtasim, Veeral
Information Storage - F	Arun, Ayaz
Look and Feel - NF	Mubtasim
Usability and Humanity - NF	Ayaz, Mubtasim
Performance and Speed - NF	Veeral
Security and Privacy - NF	Arun
Legal - NF	Arun
Comprehensive System Testing - F	Veeral

Key:
F - Function
NF - Non-Functional
Red - Newly added. Was an open issue when making SRS

3. Workflow Plan

The Waterfall method is selected due to the well-defined scope and clear objectives of the project. In the case of our expense sharing application, the core features and functionality are established, allowing for a structured and sequential approach. This method provides a systematic, step-by-step progression through requirements gathering, design, development, testing, and deployment, which can be manageable for a small team with limited

resources. Comprehensive documentation at each phase is emphasized, which is advantageous when dealing with sensitive financial and personal data, as it ensures detailed records of design decisions and security measures. The Waterfall model also offers distinct milestones and deadlines for tracking progress, which aligns with our goal of completing the project by April 2024. Moreover, it discourages scope changes after a phase is completed, helping to manage scope creep effectively. However, it's essential to be cautious of limited adaptability to changing requirements and minimal user involvement during development in the Waterfall model. To succeed, effective planning, comprehensive documentation, and strict adherence to defined project phases are crucial, given our project's characteristics and constraints.

We have chosen to use GitLab as our version control system to effectively manage the project's source code. Our branching methodology is founded on a feature-driven approach, ensuring that the main branch consistently represents a fully functional iteration of the codebase. Under this approach, developers will clone the main branch and establish dedicated branches for individual features or components. The assigned developers will work on their respective branches, ensuring that a specific feature is fully developed and rigorously tested before integration into the main branch. In this manner, the main branch remains a pristine and fully operational representation of our project. This approach embodies a structured and disciplined methodology that upholds code integrity and reliability within our project.

1. Clone the Main Branch:
    a. Initiate the process by creating a clone of the primary branch, ensuring that you have the latest and most stable version of the codebase.
2. Create a New Branch (Named After the Feature):
    a. Establish a new branch, and assign it a name reflective of the specific feature or functionality you are planning to develop.
3. Work on the Feature:
    a. Dedicate efforts to the development of the designated feature, adhering to coding standards and project requirements.
4. Test and Debug:
    a. Conduct rigorous testing and debugging to identify and rectify any issues or errors in the feature's implementation.
5. Push the Feature Branch to the Main Branch:
    a. Following thorough testing and debugging, push the feature branch to the main branch to integrate the newly developed feature into the primary codebase

We have opted to utilize Amazon Web Services (AWS) for storing our data within a secure SQL database environment. AWS offers a highly reliable and scalable platform for database management. Our data is stored in Amazon RDS (Relational Database Service), which provides us with a fully managed SQL database solution, eliminating the need for database administration tasks. By leveraging AWS RDS, we ensure that our data is stored securely, with features like automated backups and encryption. This not only enhances data integrity but also allows for easy scalability, enabling us to accommodate increasing data loads as our application grows. AWS's SQL database solution provides a foundation for data consistency and efficiency, aligning seamlessly with our project's requirements and objectives.

There are a range of tools/methods we can use to achieve performance metrics in our SRS. Database Management Systems (DBMS) such as MySQL are essential for data storage, while front-end frameworks like React or Angular aid in UI development. We can utilize expense tracking libraries and algorithms for accurate expense management, and integrate notification services and scheduling tools for timely notifications. Localization and currency support rely on i18n libraries and currency conversion APIs. We can also use things like surveys and feedback tools to retrieve metrics about how users feel about the app. By implementing automated testing with tools like selenium, and employing CI/CD pipelines we can get continuous integration and deployment. Project management and collaboration can benefit from tools like GitLab for version control, and user support and feedback can be facilitated with helpdesk software. Error tracking, user data analytics, documentation systems, and knowledge sharing are integral for ongoing improvements and decision-making. By utilizing these tools and methods, we can work toward meeting and exceeding the applications performance goals and ensuring a seamless user experience.

4. Proof of Concept Demonstration Plan

In the proof of concept demonstration, we aim to present the foundational features of our application through a mock front-end, complemented by a simplified backend. This approach serves to illustrate the core concepts and capabilities of our project. By employing the mock front-end, we intend to provide a visual guide to the application's potential, showcasing its functionalities and how users can seamlessly interact with the interface. Our primary objective in this demonstration is to ensure a user-friendly experience, where users can intuitively navigate the mock website, create expense-sharing groups, manage financial data, and more.

While the primary emphasis is on the mock front-end, we will also offer a glimpse of the supporting limited backend, encompassing database management and user roles that align with front-end interactions. This inclusion of a light backend component serves to highlight the relationship between the front-end and back-end aspects of our application. Through the execution of these strategies, we aim to gain valuable insights into the feasibility and effectiveness of our financial application.

5. Technology

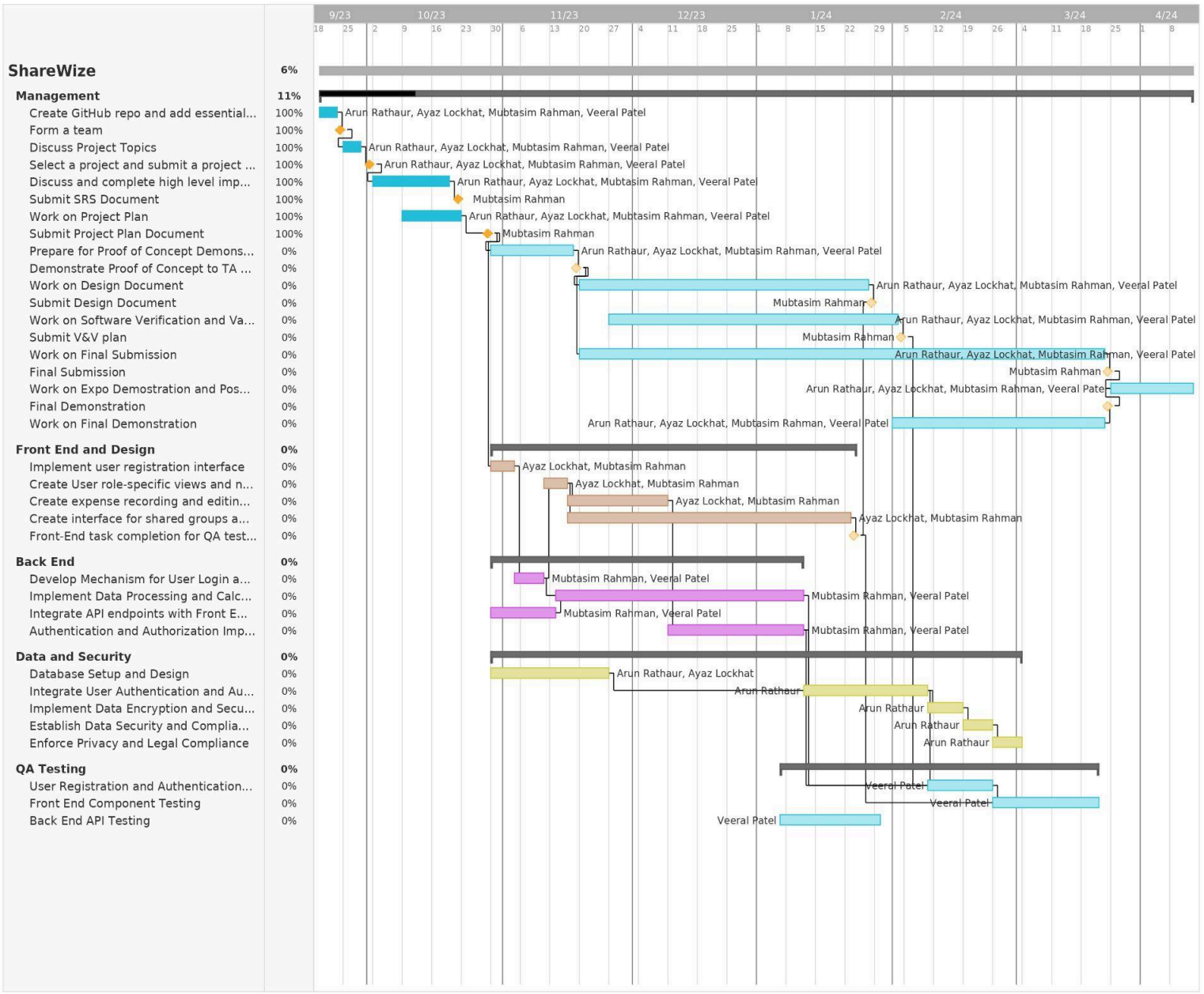
- **Coding Environment**: Visual Studio Code will be used as the default code editor throughout the project.
- **Front-End Library and Language**: We will adopt React and Bootstrap as our front-end library and TypeScript as our main programming language. React's component-based architecture enhances code reusability and user interface development. TypeScript, a statically typed superset of JavaScript provides better code readability and maintainability over vanilla JavaScript thus future-proofing our implementation. This combination ensures that our front-end development remains efficient and scalable over time.
- **CSS Frameworks**: We will use Bootstrap for its responsive capabilities which will allow our app to be used on different screen sizes and AntDesign for a modern and sleek design signature.
- **Back-End Library and Language**: We will leverage Node.js, Express.js, Sequelize, and AWS to build and manage our application, ensuring a dynamic and secure platform.
- **Database Solution:** We will AWS's Cloud Computing platform.
- **Unit testing:** We will use Puppeteer for end-to-end testing and jest for Typescript . We will use unit testing because it will allow us to find any bugs in our code early in the process by ensuring each individual part of the code works in isolation. This will create quality assurance. Since this is also a group project, it will make sure all the code with different features works together cohesively.
- **Continuous Integration:** Will be done Via GitLab's CI/CD features.

6. Coding Standard

In our development plan, we'll establish coding standards to ensure consistency and maintainability across our technology stack. We will follow widely-recognized style guides and tools, such as Airbnb JavaScript Style Guide for React.js, TypeScript TSLint or ESLint with Airbnb Style Guide for TypeScript, ESLint with Airbnb Style Guide for Node.js and Express.js. These

standards will cover code formatting, naming conventions, and documentation practices. We'll also integrate regular code reviews to enforce the proper following of these standards, fostering collaboration and code quality.

7. Project Scheduling



Attached Above is the Gantt Chart for our project which details a tentative Schedule