**Software Process Selection and Project Plan**

**<Project Code>:<Project Name>**

**<team member names & ids>**

|  |  |
| --- | --- |
| **Student ID** | **Name** |
| **Hafsa Ahmed** | **24100308** |
| **Abdul Muiz** | **24100175** |
| **Bisma Nawaz** | **24100277** |
| **Waleed Nadeem** | **24100282** |
| **Abdur Rafae Haroon** | **24100127** |

|  |  |  |
| --- | --- | --- |
| **Content** | **Totals** | **Obtained** |
| Software Process Selection | 25 | 25 |
| Project context analysis | 10 | 7 |
| Gantt chart | 25 | 15 |
| Development environment preparation | 20 | 20 |
| Deployment platform | 10 | 10 |
| Who did what | 3 | 3 |
| Review checklist | 2 | 2 |
| Overall formatting/template | 5 | 5 |
| Late submission penalty | -20 |  |
| **Total** | **100** | **87** |
| Review | 20 |  |
| **Grand Total** |  |  |

Risk Management: 20/20

**Table of Contents**

[1.](#_gjdgxs) Introduction 3

[2.](#_30j0zll) Software Process Selection 4

[3.](#_1fob9te) Gantt Chart 5

[4.](#_1t3h5sf) Risk Management 5

[4.1 Potential Risks and Mitigation Strategies 5](#_4d34og8)

[5.](#_3znysh7) Development Environment Preparation 6

[6.](#_2et92p0) Deployment Platform 6

[7.](#_tyjcwt) Who Did What? 7

[8.](#_3dy6vkm) Review checklist 7

# Introduction

The proposed project is a user-hosted trivia web application. This application aims to create an interactive quiz-based learning experience, taking its inspiration from the popular game-based learning platform, ‘Kahoot!’. Designed for students, teachers, trainers or simply anyone who wants to challenge their friends to a quiz, the extent of the web application’s features allow anyone to quickly create and host a quiz-based session and let others join in and participate. Thus, within a session, there will exist a host user who creates the trivia while the participants joining voluntarily act as players. The players will use the quiz ID given by the host to join a particular session.

# Software Process Selection

**2.1**

Waterfall and Agile (Scrum) are two distinct software development methodologies, each with its own set of advantages and disadvantages.

**Waterfall:**

**Pros:**

* **Structured Approach:** Waterfall follows a clear and sequential structure, making it easier to plan and manage.
* **Well-Defined Requirements:** It works well when the project requirements are well-understood and unlikely to change.
* **Documentation:** Extensive documentation at each stage ensures a clear record of the project's progress.

**Cons:**

* **Rigidity:** The rigid structure of Waterfall can be problematic when requirements evolve or change during the project.
* **Limited Client Involvement:** Clients often have limited input until the final stage, which can lead to unexpected outcomes.
* **Testing at the End:** Testing is typically done after the development phase, potentially leading to issues being discovered late in the process.

**Agile (Scrum):**

**Pros:**

* **Flexibility:** Agile allows for changes and iterations throughout the project, making it suitable for dynamic or evolving requirements.
* **Client Collaboration:** Clients have continuous involvement, providing feedback and ensuring the final product aligns with their vision.
* **Early Deliveries:** The project can deliver incremental results, offering value to the client sooner.

**Cons:**

* **Complex to Manage:** Agile projects require effective team collaboration and management.
* **Lack of Predictability:** The flexible nature of Agile can make it challenging to estimate the final project timeline and cost.
* **Documentation May Be Limited:** Agile prioritizes working software over comprehensive documentation.

**2.2**

For a user-hosted trivia web application, an Agile (Scrum) approach would be better suited.

**2.3**

Justification for choosing Agile (Scrum) for this project:

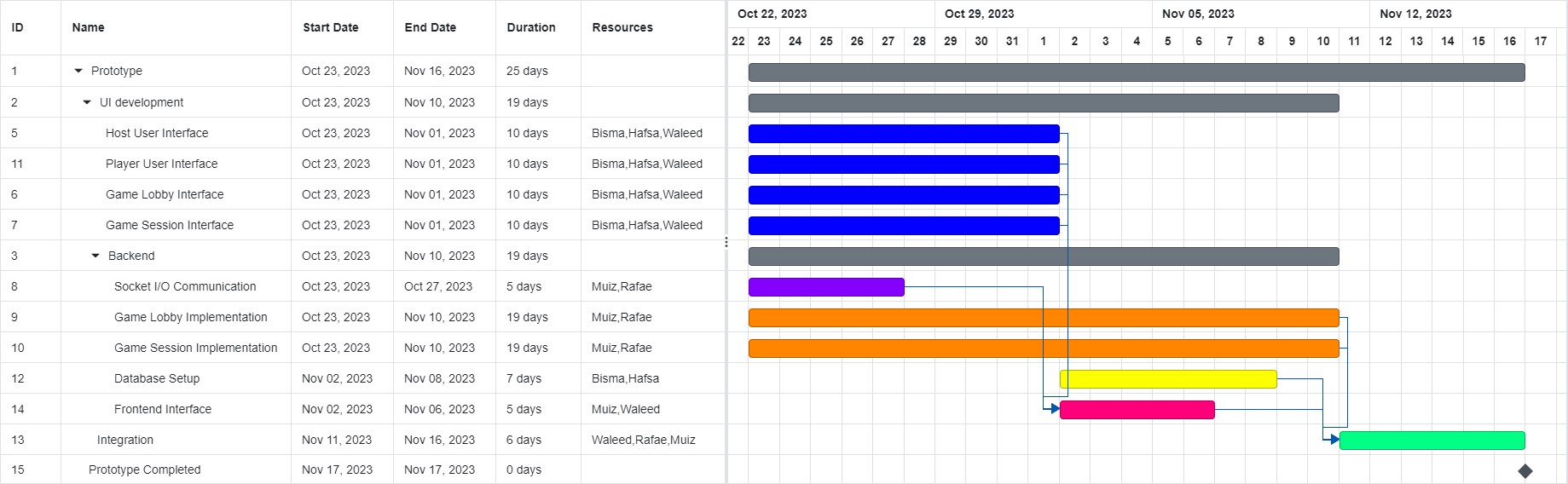
* **Project Context Analysis:** In the context of a user-hosted trivia web application, the requirements are likely to evolve as user feedback is gathered. New trivia questions may need to be added, existing questions modified, and new features incorporated based on user preferences. This dynamic and interactive nature of the application aligns well with Agile principles, which allow for flexibility and iterative development.
* **User Involvement:** Agile encourages continuous client (or in this case, user) involvement, which is crucial in developing a user-centric application. In this project, user feedback will be invaluable in improving the trivia content and user experience.
* **Early Deliveries:** Agile enables the delivery of incremental results, allowing the application to go live with essential features quickly. This will provide users with a usable product early on, which is especially important for a web application designed for interaction and engagement. For this project, prototype is delivered as early deliverable.]
* **Iterative Development:** Agile allows the team to iterate on the application, making improvements and adjustments as it progresses. This is vital for a trivia application as it can continuously evolve and adapt to user preferences and trends.

In summary, the Agile (Scrum) methodology is chosen for this project due to its flexibility, user-centric approach, and suitability for a web application that requires constant adaptation and user feedback to provide an engaging and interactive experience.

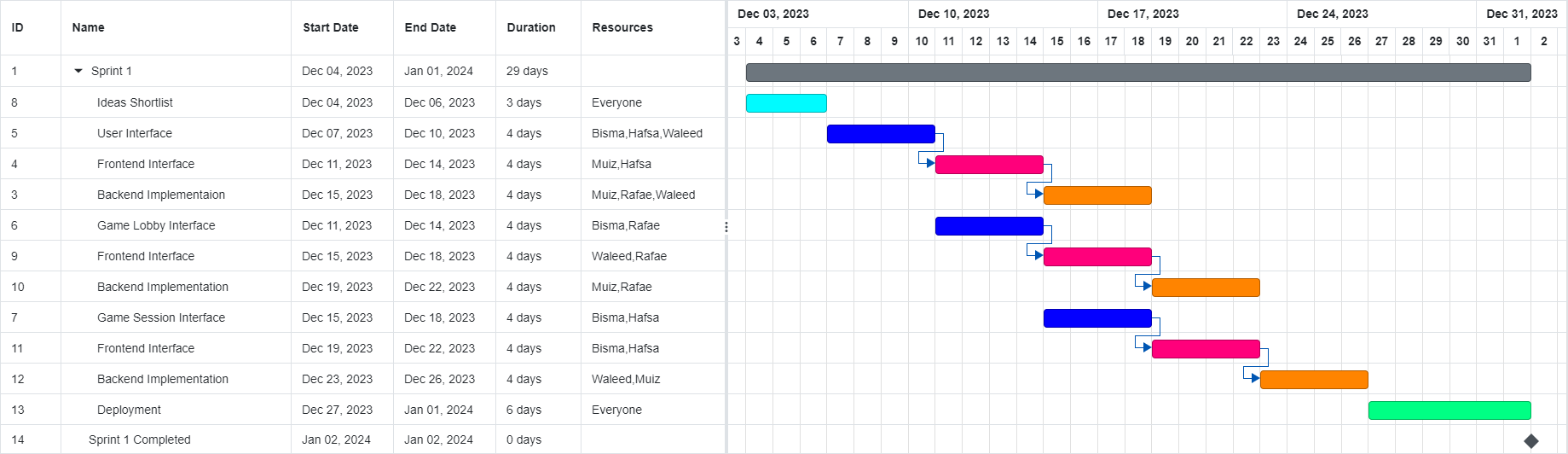
# Gantt Chart

[Missing plan for sprints 2,3,4 and final integration testing. You also need to identify the role of the individual team members in the plan, not group of 2 or 3 members always.]

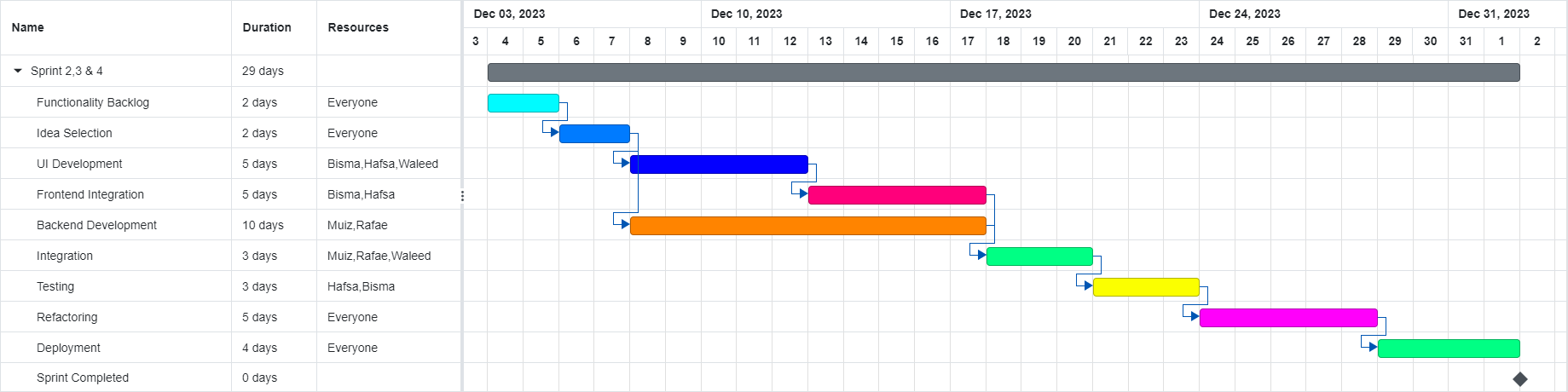
**Prototype**

[****](https://drive.google.com/file/d/1_AeXw3k-h7ZVIefIFsVGdjV1R9PQYxei/view?usp=sharing)

**Sprint 1**

[](https://drive.google.com/file/d/1j6Fr5IsIJ7is8FIxk-wzWNthCXPxGU74/view?usp=share_link)

**Sprint 2,3,4**

[****](https://drive.google.com/file/d/1slXCixoViP20HY3SMhuDluaciPf_aRUT/view?usp=sharing)

Ignore dates, sprints 2,3 & 4 follow similar general tasks

# Risk Management

**4.1 Potential Risks and Mitigation Strategies**

|  |  |  |
| --- | --- | --- |
| **Sr.** | **Risk Description** | **Mitigation Strategy** |
|  | **Inadequate server scalability:** The server infrastructure may not be able to handle sudden spike in traffic (requests), increasing latency or even leading to server crashes. | Implement auto-rescaling mechanisms and regularly monitor server performance with traffic to ensure it can handle increased load. |
|  | **Third-party dependency:**  Dependence on third-party services (supabase) can lead to disruptions if these services face sudden outages. | Have backup plans: data backups, and consider alternative providers. Design while keeping the risk of these dependencies in mind from the start. |
|  | **Underestimated development time:**  Underestimation of the time required for the development can lead to project delays. | Use structured project estimation techniques, track progress regularly, and consider using pre-built components to save some time. |
|  | **Key member unavailability:**  Team members falling ill or becoming unavailable during important project phases due to unavoidable circumstances. | Documenting project progress, processes, procedures, and all other useful information comprehensively.  Encouraging all team members to take on roles across different areas, so they are equipped to take on someone else’s tasks in case of emergency.  Ensuring clear communication of tasks and deadlines ahead of time, and incorporating hybrid work settings, if the task allows.  Maintaining a shared centralized github repository to ensure that each team member fully understands the project needs, and can fill the gap of a missing member if need be. |
| 5. | **Data loss:**  Inadequate data backup and recovery mechanisms can lead to data loss in the case of server failures. | Implement regular automated data backups for all critical data and test the data recovery process to ensure data integrity and availability.  Ensure that access to the data is limited to authorized users by putting access controls in place.  Store sensitive data in the form of offline backups such as physical storage. |
| 6. | **Integration challenges:**  Tools used in the project may not work seamlessly together, leading to integration problems. | Perform thorough compatibility testing and select tools that work best together. |
| 7. | **Performance issues:**  Inefficient code, poorly optimized database, or inefficiently scaled resources can lead to performance issues and increased load times. | Conduct load and stress testing to identify specific pain points and improve accordingly.  Review codebase for optimality by striving for more efficient algorithms.  Optimize database queries to retrieve data in minimal time, leading to reduced latency and a much better user experience. |
| 8. | **Dependency failures:**  Reliance on external libraries, APIs, or services may pose risks if these dependencies experience issues or changes. | Keep a comprehensive inventory of all dependencies used, and check for updates.  Use Git to rollback to previous states in case of dependency failures or lock dependencies to specific versions in the project’s files where possible to reduce risk of dependency issues.  Make use of a diverse set of dependencies to counter reliance on any one of them.  Rigorous testing in the development phase can also help identify dependency issues early on. |
| 9. | **Lack of user engagement:**  Users may not engage with the application as expected, thus resulting in lower user interactions. | Regularly gather user feedback, and implement user engagement strategies such as notifications to encourage user engagement, and increase impressions. |
| 10. | **Security vulnerability:**  Failure to address security concerns raised by the users and certain vulnerabilities can result in unauthorized access or data breaches. | Implement better security practices by ensuring strong data encryption, conduct regular security audits , and try to keep all the libraries up to date. |

# Development Environment Preparation

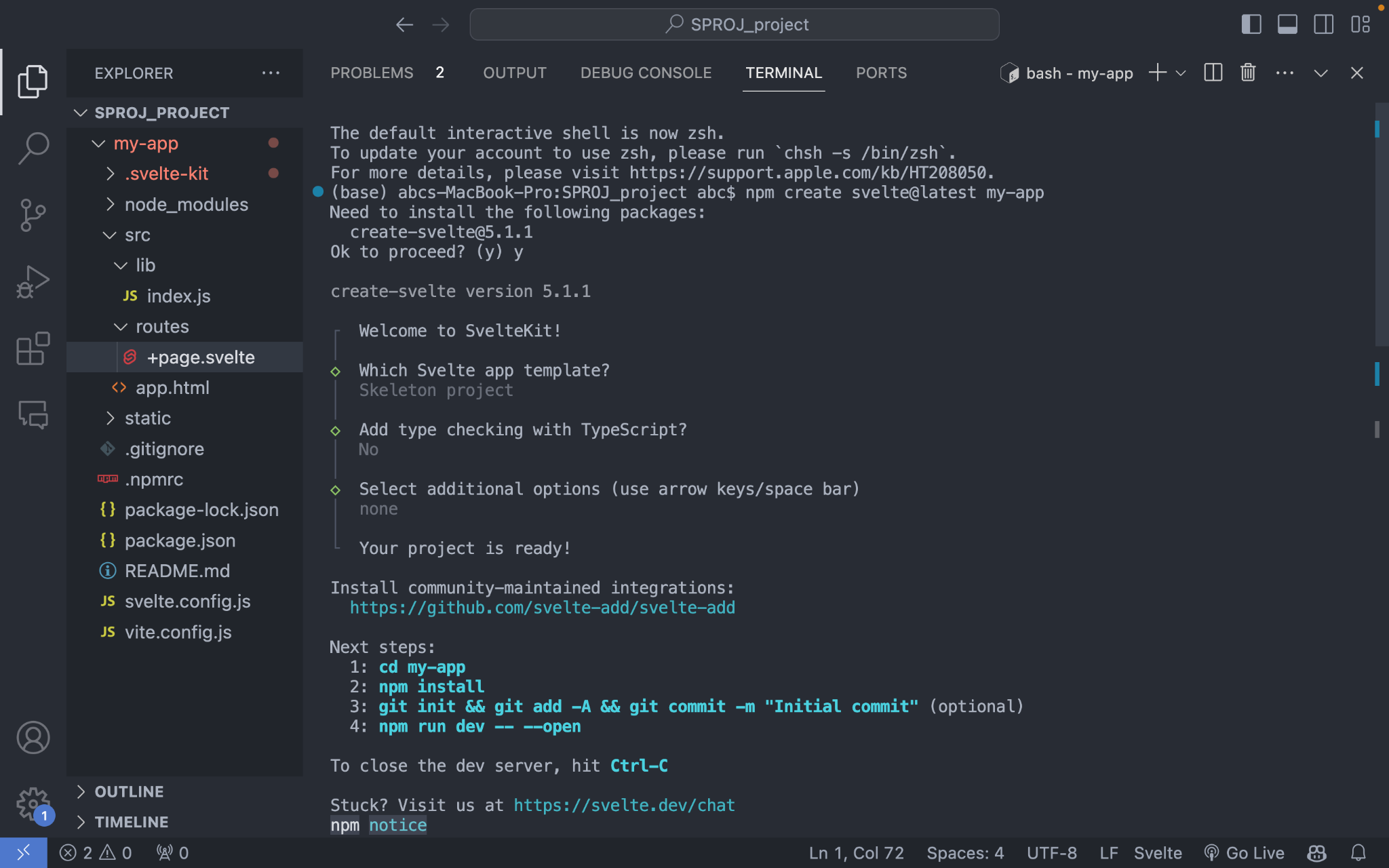
**Tools and Technologies**

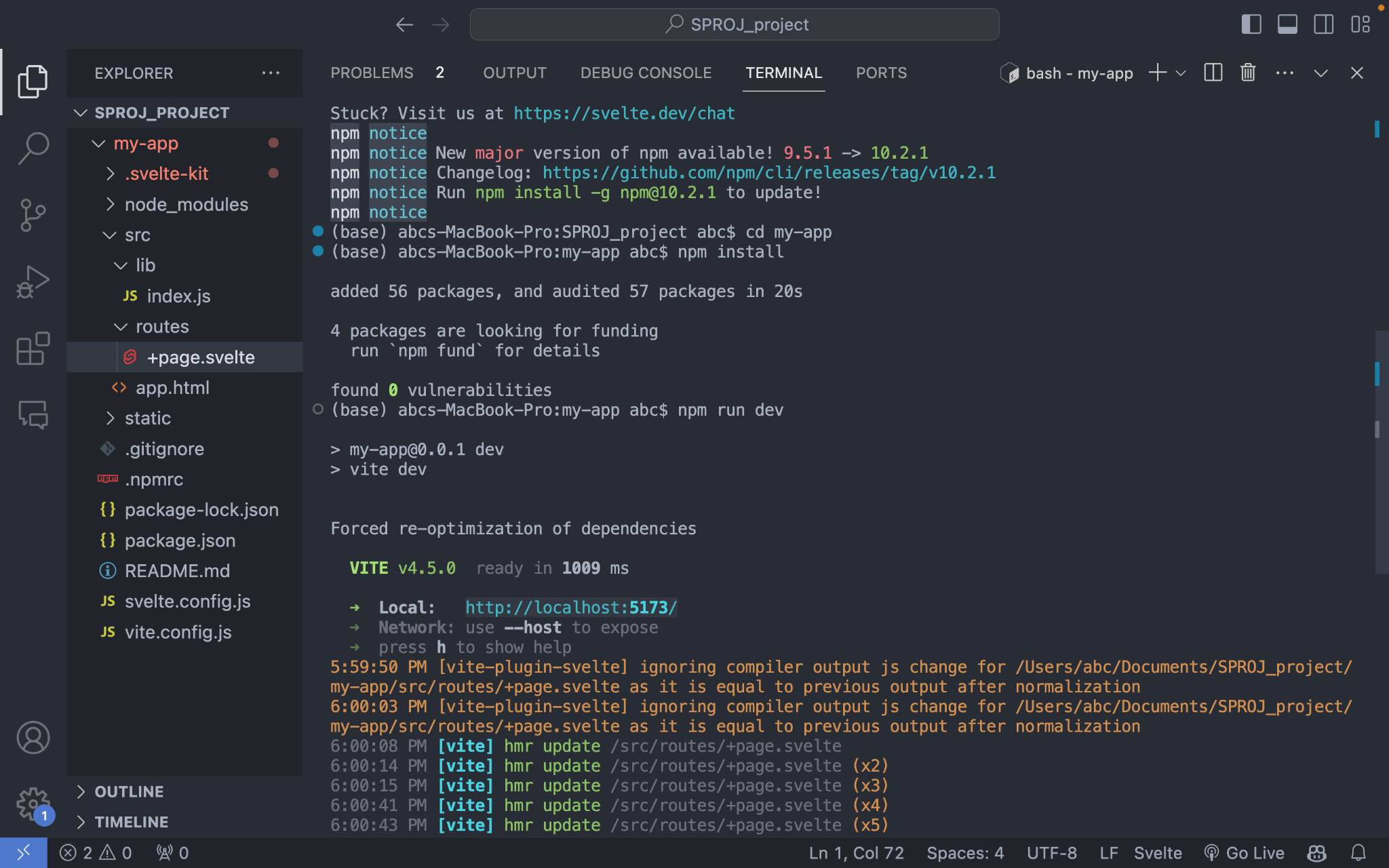
* SvelteKit (Front-end and Back-end)
* Visual Studio Code (Code editor)

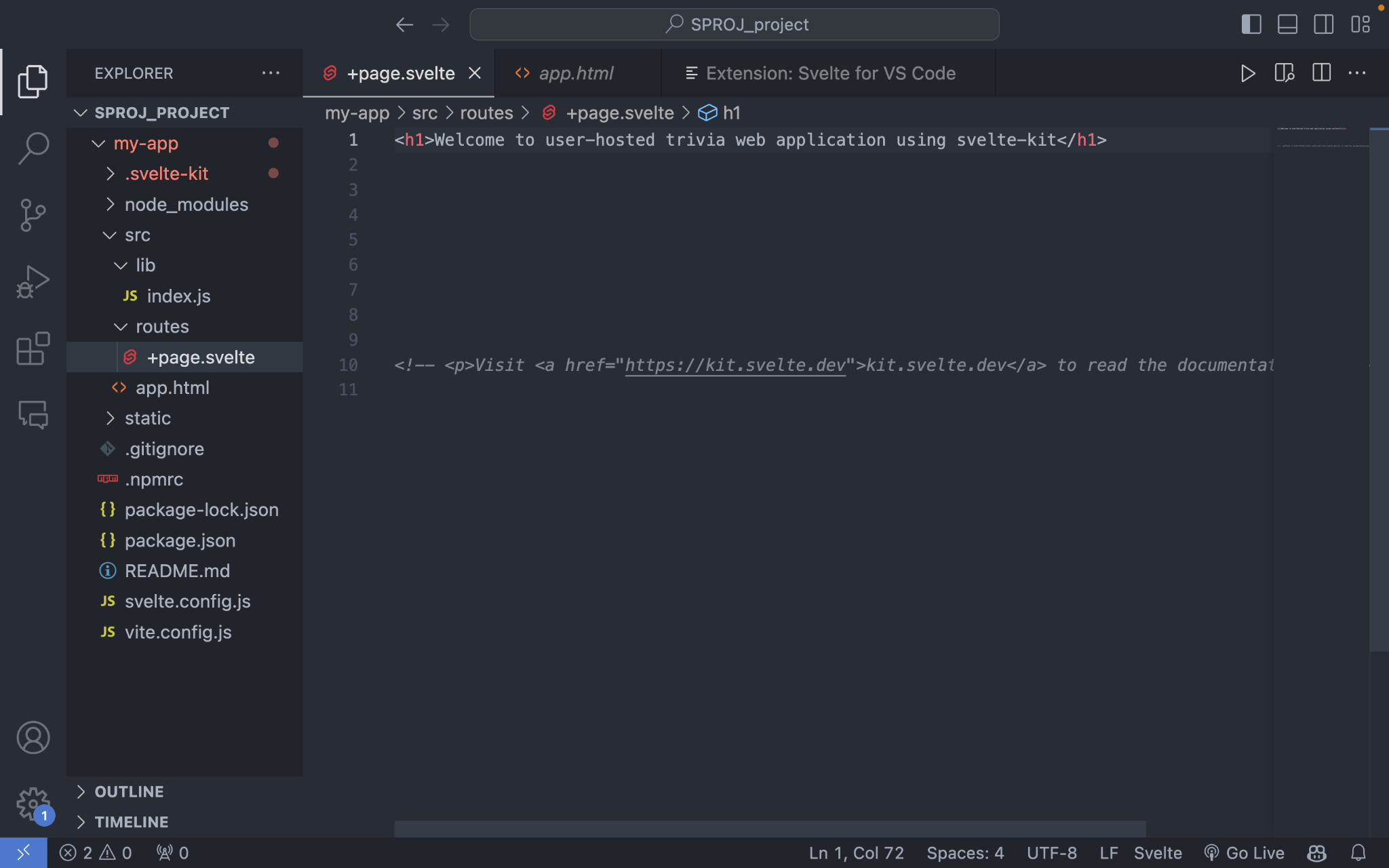
**Development Environment**

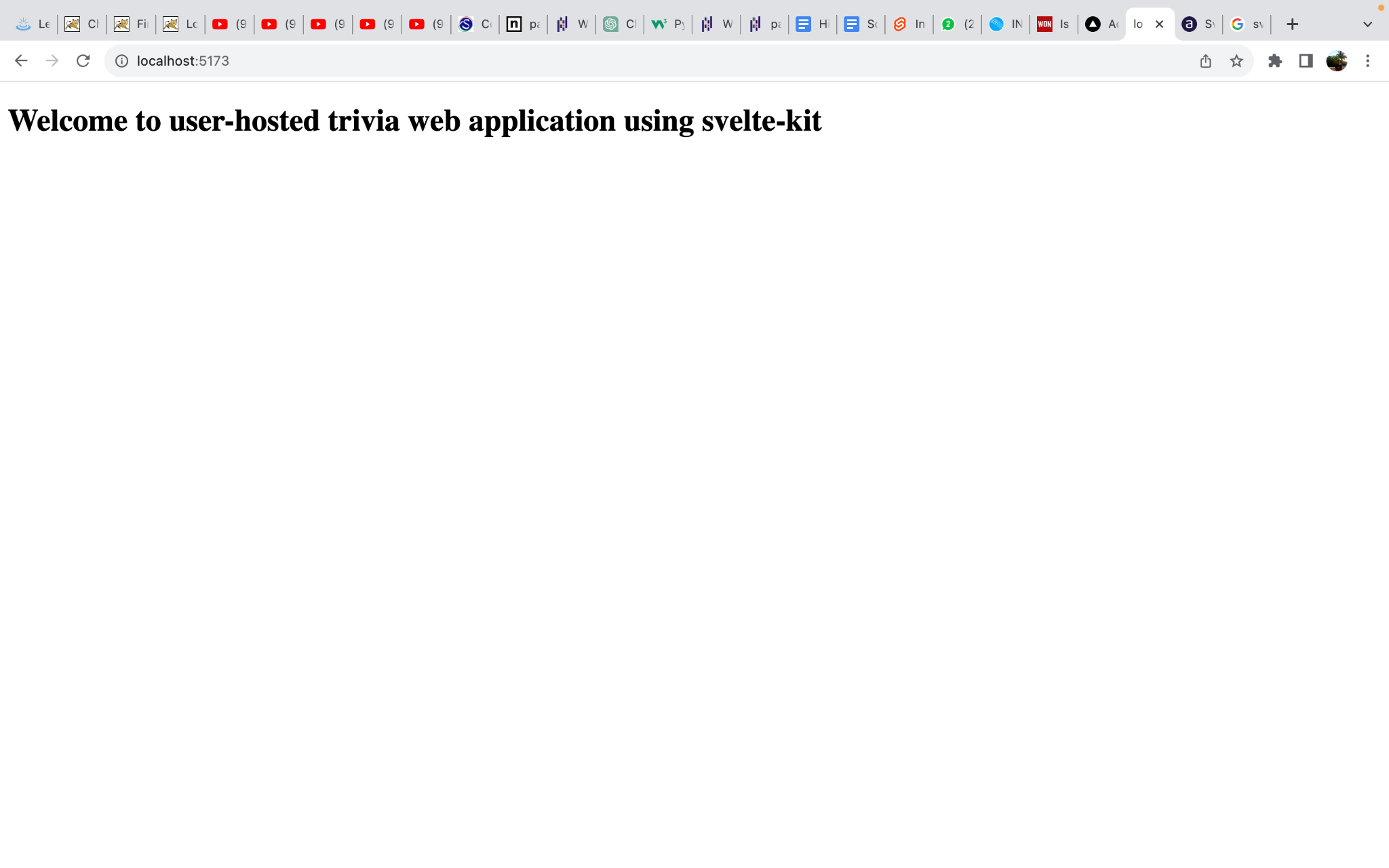
* Development Environment has been set up
* Sveltekit environment has been set up on VS Code with the help of SvelteKit docs
* A default landing page has been set up that welcomes users to the web-hosted trivia application.
* The website for the time-being has been deployed locally to localhost:5173.
* Screenshots are attached below

**Screenshots**

****

****

****

****

# Deployment Platform

**Vercel**

* We will be using Vercels free tier called Hobby for our prototype deployment
* It provides 100 Gb of bandwidth along with one hour of run-time logs which should be enough for our prototype.
* It provides automatic scalability as well as supports server-less functions. This is helpful as svelte-kit routes are bundled into a single server-less function which can easily be deployed to Vercel.

# Who Did What?

|  |  |
| --- | --- |
| **Name of the Team Member** | **Tasks done** |
| Bisma Nawaz | Risk management |
| Hafsa Ahmed | Risk management (Mitigation Strategies) |
| Waleed Nadeem | Software Process Selection |
| Abdul Muiz | Gantt Chart |
| Abdur Rafae Haroon | Development Environment Preparation and Deployment Platform |

# Review checklist

Before submission of this deliverable, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Section** **Title** | **Reviewer Name(s)** |
| 3 | Bisma Nawaz |
| 2 | Hafsa Ahmed |
| 4 | Waleed Nadeem |
| 2,4,5,6 | Abdul Muiz |
| 4 | Abdur Rafae Haroon |