# Project Plan

***Real-Time Event Dashboard***

*BAS World*

| | | |
|---|---|---|
| **Date** | **:** | **10th September** |
| **Version** | **:** | **1.1** |
| **State** | **:** | **On-going** |
| **Author(s)** | **:** | **Jordi Blaas, Lisa Diepstraten, Anh Huynh, Alperen Korkmaz, Renato da Silva Pereira, Aleksandar Stoynov, Rositsa Mazhlenkova** |

**Version history**

| Version | Date | Author(s) | Changes | State |
|---|---|---|---|---|
| 0.1 | 10/9/2024 | Jordi Blaas, Lisa Diepstraten, Anh Huynh, Alperen Korkmaz, Renato da Silva Pereira, Aleksandar Stoynov, Rositsa Mazhlenkova | An initial first draft of the project plan | Draft |
| 1.0 | 16/9/2024 | Jordi Blaas, Lisa Diepstraten, Anh Huynh, Alperen Korkmaz, Renato da Silva Pereira, Aleksandar Stoynov, Rositsa Mazhlenkova | Adjust the project plan based on the tutor’s feedback | On-going |
| 1.1 | 21/10/2024 | All authors | Adjust the project plan based on the PO’s feedback | On-going |

**Distribution**

| Version | Date | Receivers |
|---|---|---|
| 0.1 | 16/9/2024 | Raja Gorentla (Instructor) |
| 1.0 | 23/9/2024 | Raja Gorentla (Instructor) |
| 1.1 | 21/10/2024 | Raja Gorentla (Instructor) and Rob Eijgelshoven (PO) |

## Contents

# 1. Project assignment

## 1.1 Context

BAS World is a global marketplace specializing in buying and selling commercial vehicles, equipment, and machinery. As part of its digital strategy, BAS World has implemented an event generation system on its website that captures user interactions, such as clicks, form submissions, and other actions. These events are accompanied by metadata, including timestamps and user identifiers, which provide valuable insights into how users engage with the platform.

## 1.2 Problem statement

| **Problems** | **Solutions** |
|---|---|
| **Absence of Centralized Event Monitoring System:** BAS World lacks a centralized system to capture and display real-time event data, making it difficult to track user actions and understand their behavior. | **Develop a Real-Time Event Dashboard:** Create a centralized system that processes, stores, and displays website events in real-time to provide BAS World with actionable insights. |
| **Lack of Real-Time Insights**: Without a real-time dashboard, BAS World cannot monitor user interactions as they occur, leading to delayed decision-making and missed opportunities for optimization. | **Implement Real-Time Data Access**: Design and develop a real-time dashboard that provides BAS World with immediate visibility into user actions for faster, data-driven decision-making. |
| **Data Organization and Storage Challenges**: BAS World lacks a structured system to store and organize event data, making it difficult to retrieve and analyze historical data efficiently. | **API and Back-End System for Data Processing**: Develop an API and back-end system to organize, validate, and store event data in a structured format for both real-time and historical analysis. |

| | |
|---|---|
| **Complexity of Visualizing Event Data**: BAS World has no visual interface to represent event data, making it hard to gain insights from user interactions. | **Front-End Dashboard for Data Visualization**: Design a user-friendly, web-based dashboard that displays real-time event data with clear visualizations, making it easy to track and analyze user behavior. |
| **Scalability and Deployment Limitations**: As the volume of event data grows, BAS World will require a scalable solution capable of handling increased traffic and data. | **Scalable Architecture with Optional Cloud Deployment**: Build a scalable system and optionally deploy using Docker and AWS (SNS/SQS) for flexibility, ensuring the system can grow with BAS World's needs. |

## 1.3 Goal of the project

The goal of this project is to create a real-time event dashboard that will enable BAS World to process and display user interactions on their website. By building an API to handle event data, a back-end system for storage, and a front-end dashboard for real-time visualization, BAS World will gain valuable insights into user behavior. This will improve decision-making, optimize the user experience, and enhance business performance. Additionally, the project offers the potential for scalable deployment using Docker and AWS, ensuring the system can grow with the company's needs.

## 1.4 Scope and preconditions

| **Inside scope:** | **Outside scope:** |
|---|---|
| Creating a real-time dashboard for BAS World to visualize event data and enable real-time monitoring of user interactions. | The creation of an active event queue by BAS World (simulated by the team). |
| Developing an API to receive simulated event data and process it into the back-end system. | Extensive front-end design beyond the necessary real-time data presentation. |

| | |
|---|---|
| Building a back-end system to store and organize event data securely and efficiently. | Implementation of security mechanisms such as encryption, user accounts, roles, or access control. |
| Simulating an event queue for testing, since the actual event generation system is not yet operational. | |
| Implementing basic front-end design to display real-time data with a functional user interface. | |

- **Data Schema Provided by BAS World**: The team must use the data schema provided by BAS World for generating and processing event data.
- **Technology Choices**: The team has the autonomy to select the technology stack, but all choices must be justified based on scalability, functionality, and performance.
- **Agile Methodology**: The project will follow Agile development practices, with regular feedback from the Product Owner.
- **Critical Mindset**: The team is expected to maintain a critical but constructive approach toward pre-existing decisions and technologies, ensuring they are suitable for the project's goals.

## 1.5 Strategy

The project will follow an Agile methodology, specifically Scrum, due to its iterative nature and frequent feedback loops. This approach allows the team to adjust quickly based on feedback from the Product Owner and ensures alignment with business needs. The project will run through five sprints, each with defined deliverables and milestones.

## 1.6 Research questions and methodology

**Research Questions**

1. **What is the best approach for simulating event data for BAS World's website?**
   - **Methodology**: Research will involve exploring data simulation techniques commonly used for websites with high volumes of interactions. Literature

reviews and tool evaluations (like event stubs or queue systems) will help determine the best practice for realistic simulations.

2. **How can event data be processed and stored efficiently to support real-time analysis?**
    - **Methodology**: This will involve investigating various database systems (SQL vs NoSQL) and messaging services (such as Kafka, RabbitMQ, or AWS SNS/SQS). Experiments and benchmarking will be performed to compare their efficiency and scalability.
3. **What are the most effective ways to visually represent real-time event data on a dashboard?**
    - **Methodology**: Front-end design research will focus on user experience (UX) principles, analyzing successful dashboards in similar domains (e.g., e-commerce). Design tools like Figma or Adobe XD may be used to prototype different dashboard layouts before implementation.

**Methodology**

The project will follow a **Dot Framework** methodology to organize research across various contexts (Workshop, Lab). During **Workshop** phases, findings will be discussed with stakeholders to refine deliverables and during **Lab** phases, experimental setups will validate assumptions (e.g., testing data processing solutions).

Research will be iterative and integrated into each project phase. As the project progresses, new research questions may emerge, and the approach will be adapted accordingly.

## 1.7 Deliverables and non-deliverables

| **Deliverables** | **Non-Deliverables** |
|---|---|
| **Simulated Event Data System**: A tool or stub to generate and simulate event data for testing purposes. | **Advanced Front-End Design**: The system will focus on functionality rather than extensive front-end styling or design. |
| **API Service**: A back-end API that processes and validates event data | **Real-Time Queue Creation by BAS World**: The actual event queue will be |

| | |
|---|---|
| received from the simulation system or real queue. | provided or handled by BAS World. The team will simulate the queue for the project. |
| **Back-End Database**: A structured database that securely stores and organizes event data for real-time and historical analysis. | **Third-Party Integrations**: Integrations with external tools or platforms (beyond optional AWS deployment) are not included. |
| **Front-End Dashboard**: A web-based dashboard that visually represents event data in real-time with filtering and report generation functionalities. | **Full Deployment on AWS (Optional)**: While optional, the deployment of the system using Docker and AWS (SNS/SQS) may be attempted if time allows but is not mandatory. |
| **Architecture Documentation**: Complete technical documentation outlining system architecture, components, and processes. | |
| **Testing Reports:** Unit, integration, and system testing reports to ensure the system meets functional and non-functional requirements. | |
| **Final Project Presentation:** A comprehensive final presentation summarizing the project outcomes, architecture, and learnings. | |
| **Docker Container:** A containerized version of the system, allowing easier testing, deployment, and scalability. | |

# 2. Project Organization

## 2.1 Stakeholders and team members

| Name | Abbreviation | Role and functions | Availability |
|---|---|---|---|
| Rob Eijgelshoven (Product Owner, BAS World) | *R.E.* | Product Owner (BAS World) - Provides feedback and direction on requirements. | Communication with the Product owner will be initiated by the team. There are scheduled meetings at the end of each sprint. |
| Raja Gorentla (Instructor, Fontys University) | *R.G.* | Instructor (Fontys University) - Supports academic requirements and ensures project success. Acts as the source of communication with the product owner. | Monday (09:00 - 12:00), Tuesday (13:00 - 16:00) |
| Lisa Diepstraten | *L.D.* | Group Member | Monday, Tuesday, Wednesday |
| Anh Huynh | *A.H.* | Group Member | Monday, Tuesday, Wednesday |
| Alperen Korkmaz | *A.K.* | Group Member | Monday, Tuesday, Wednesday |
| Renato da Silva Pereira | *R.S.P.* | Group Member | Monday, Tuesday, Wednesday |
| Jordi Blaas | *J.B.* | Group Member | Monday, Tuesday, Wednesday |
| Aleksandar Stoynov | *A.S.* | Group Member | Monday, Tuesday, Wednesday |

| | | | |
|---|---|---|---|
| Rositsa Mazhlenkova | *R.M.* | Group Member | Monday, Tuesday, Wednesday |

## 2.2 Communication

- Sprint Reviews: Held every three weeks on Tuesday afternoons with the Product Owner to review progress and gather feedback at the end of each sprint.
- Daily Stand-Ups: Short, daily team meetings to track progress, discuss blockers, and set daily goals.
- Communication Channels: Teams, Whatsapp, E-mail.
- Weekly Tutor Meetings: Regular meetings with the tutor to receive guidance, clarify requirements, and ensure alignment with academic expectations.
- Sprint Presentations: At the end of each sprint, a formal presentation showcases completed work, followed by a planning session for the next sprint.
- Final Presentation: A comprehensive presentation delivered at the end of the project, summarizing outcomes, key challenges, and overall achievements.

# 3. Activities and time plan

## 3.1 Phases of the project

### Phase 1: Preparation (Sprint A - Completed)

**Tasks:**

- Create project documents, initial backlog, and select the technology stack.
- Prepare initial API endpoints and dashboard prototype.

**Status:** Completed

**Deliverables:**

- Finished project documents.
- Initial API endpoints created.
- Dashboard front-end prototype built.

- Sprint A presentation, including diagrams.
- Simulated event data created.

**Sprint A Review:**

- Presented initial event simulation and project structure.
- Received feedback from the Product Owner (PO).

---

## Phase 2: Event Data Simulation and Basic Functionality (Sprint B)

**Tasks:**

- Simulate event data.
- Focus on core API development for receiving and processing events.
- Implement basic data storage.

**Deliverables:**

- Fully functional API capable of receiving and processing simulated event data.
- Data validation and storage in a database.
- Real-time data simulation is integrated with the dashboard to allow PO feedback.

**Sprint B Milestones:**

- Event simulation is displayed on the dashboard.
- Core API for event processing completed.

**Feedback Goal:**

- Validate event simulation accuracy and API structure with the PO, ensuring early functionality.

**Corresponding User Stories:**

- US1 (Real-Time Event Feed) – Must Have: Priority 100.
- US2 (View Detailed Event Information) – Must Have: Priority 95.

## Phase 3: API Expansion, Data Processing, and Security (Sprint C)

**Tasks:**

- Expand API functionality with filtering options (event type, date range).
- Implement data security measures such as HTTPS for secure transmission.

**Deliverables:**

- Full API functionality with filtering options.
- Secure data transmission with encryption (HTTPS).
- Data stored securely.

**Sprint C Milestones:**

- Filters added to the API and dashboard for real-time event monitoring.
- Secure data processing implemented.

**Feedback Goal:**

- Validate filtering and secure data processing with the PO, ensuring the system aligns with real-world use cases.

**Corresponding User Stories:**

- US3 (Filter Events by Type and Date) – Should Have: Priority 85.
- US6 (Ensure Data Security) – Must Have: Priority 90.

---

## Phase 4: Front-End Dashboard Enhancement and Reporting (Sprint D)

**Tasks:**

- Refine the front-end dashboard, focusing on user experience (UX) and usability.
- Add detailed event reporting features.

**Deliverables:**

- Enhanced UI/UX for dashboard with real-time data visualization and filtering.
- Event reporting feature (summary of actions, peak times, user engagement).

**Sprint D Milestones:**

- Real-time data visualization and filtering.
- Event reporting feature completed.

**Feedback Goal:**

- Validate the usability of the dashboard, filtering, and reporting features with the PO for real-time monitoring.

**Corresponding User Stories:**

- **US9** (Dashboard Usability and Design) – Should Have: Priority 80.
- **US4** (Generate Reports on User Interactions) – Should Have: Priority 80.

---

## Phase 5: Scalability, Performance Testing, and Optional Deployment (Sprint E - Stretch Goal)

**Tasks:**

- Focus on performance testing under high traffic and scalability.
- Optionally deploy using Docker and AWS (if time permits).

**Deliverables:**

- Performance testing completed (system handles thousands of events with no lag).
- Docker container setup for testing and deployment.
- Optional AWS deployment (if time allows).

**Sprint E Milestones:**

- Performance and load testing completed.
- Optional deployment to AWS using Docker and SNS/SQS.

**Feedback Goal:**

- Validate the system's scalability and performance with the PO. If optional deployment is pursued, validate it in a production-like environment.

**Corresponding User Stories:**

- **US8** (Scalability and Performance) – Must Have: Priority 95.
- **Stretch Goal**: Optional containerization with Docker and deployment on AWS.

## 3.2 Time plan and milestones

| Phasing | Effort | Start date | Finish date | Outcome |
|---|---|---|---|---|
| Sprint A (Phase 1) | Documentation, Initial API | 23.09.2024 | 11.10.2024 | Completed project documentation, basic API, and dashboard |
| Sprint B (Phase 2) | Event Data Simulation, Core API | 14.10.2024 | 08.11.2024 | Fully functional API, event data displayed in real-time |
| Sprint C (Phase 3) | API Expansion, Data Filtering, Security | 11.11.2024 | 29.11.2024 | Expanded API, filtering, and data security |
| Sprint D (Phase 4) | Front-End Refinement, Reporting Features | 02.12.2024 | 20.12.2024 | Enhanced UX/UI, event reporting, and |

| | | | | |
|---|---|---|---|---|
| | | | | real-time data visualization |
| Sprint E (Phase 5) | Performance Testing, Optional Deployment | 23.12.2024 | 17.01.2025 | Performance tests completed, optional Docker and AWS deployment |

# 4. Testing strategy and configuration management

## 4.1 Testing strategy

- **Unit Testing:** Automated unit tests will be implemented at the beginning of API development to validate individual components and endpoints. These tests will be executed in Docker containers to ensure consistency across environments and prevent issues like "works on my machine". Unit tests will cover each functionality incrementally as it is developed.
- **Integration Testing:** Starting early in the development process, automated integration tests will verify interactions between services (API, database, and front-end). Docker containers will simulate the entire stack, ensuring that components interact correctly in an isolated environment. Integration tests will detect issues across components as they evolve.
- **System Testing:** Performed at key milestones to ensure that the full system operates correctly. System tests will simulate a near-production environment using Docker containers, allowing us to verify that the system performs as expected from development to production.
- **Automated Testing with Docker:** Unit, integration, and system tests will be integrated into the **CI/CD pipeline**, executed in Docker containers to guarantee reproducible

environments. Docker ensures consistency across local development, testing, and production. Tests will be automatically triggered on each code commit, continuously validating the system throughout development.

- **Early Automation**: Automated testing will be prioritized from the beginning of the development process, not delayed until the end. This approach will ensure continuous quality validation, especially when integrating components like the API and database.

## 4.2 Test environment and required resources

- **Local Development Environment**: Local environments will be used with simulated event data to test functionalities in isolation and during integration.
- **CI/CD Pipeline**: Continuous integration and continuous deployment pipelines will run tests automatically upon code commits, enabling fast feedback loops.
- **AWS**: AWS will be used for deployment testing if the stretch goal is reached.

## 4.3 Configuration management

Version control using GIT, with branching strategies to manage different features.

# 5. Finances and risk

## 5.1 Project budget

No significant financial costs are expected.

## 5.2 Risk and mitigation

| Risk | Prevention activities | Mitigation activities |
|---|---|---|
| **Data handling issues** (e.g., data loss, corruption) | Implement data validation rules and constraints at both the API and database levels. Ensure that error handling and data sanitization occur | Set up automatic data backups, versioning, and rollback capabilities to recover from data corruption or loss. Employ |

| | | |
|---|---|---|
| | during data ingestion to prevent incorrect or malicious data entries. Use data type constraints to ensure the integrity of input data. | monitoring tools to detect anomalies in data flow and trigger real-time alerts. Enable detailed error logging to identify and resolve data handling issues quickly. Regular audits and manual data testing will help ensure integrity and allow for faster fixes. |
| **Integration problems** between back-end and front-end | Define and implement integration tests that simulate end-to-end workflows, validating interactions between the API, database, and front-end UI. Begin testing as soon as initial components are functional. Integrate the front-end with the API early to catch potential communication or API issues. | Use a CI/CD pipeline to automatically run integration tests whenever a code change is made. Logs will capture any integration errors during testing. For unexpected issues post-integration, use detailed error reports and alerting systems to quickly detect and resolve problems. |
| **Unavailability of Product Owner (PO)** | Schedule regular sprint reviews and feedback sessions with the PO to ensure consistent input. Set up backup communication channels like email asynchronous communication when the PO is unavailable. Ensure | Use e-mail as the primary backup communication method if the PO is unavailable for meetings. Have the team continue work on pre-agreed priorities and make documented assumptions on decisions to avoid delays. |

| | | |
|---|---|---|
| | that requirements are clearly documented and accessible. | Conduct interim reviews based on previous feedback to keep the project aligned. |
| **Technical issues with real-time data processing.** | Implement a robust testing strategy, including scalability and performance tests. | Conduct load tests and set up fallback strategies for performance optimization. Monitor the system to detect any issues early. |
| **Data security and integrity concerns** | Follow secure API development practices such as using HTTPS for data transmission, implementing authentication and authorization, and ensuring data validation at every stage. Ensure that sensitive data is encrypted in transit and at rest. | Perform regular security audits and vulnerability scans to detect potential issues. Implement logging of security-related events (e.g., failed authentication attempts) and alerts for suspicious activities. Apply manual validation tests to check the security of the system post-integration. |
| **Team members fall sick** | Assign tasks evenly among team members so that the absence of any single team member doesn’t stall progress. Ensure all work is tracked using a task management tool (Jira) so team members can easily pick up tasks in case of illness. | Enable team members to work remotely when sick but able to work. Have shared documentation and clear project handoffs so others can temporarily take over critical tasks. For longer absences, delegate responsibilities to another team member to ensure work continuity. |