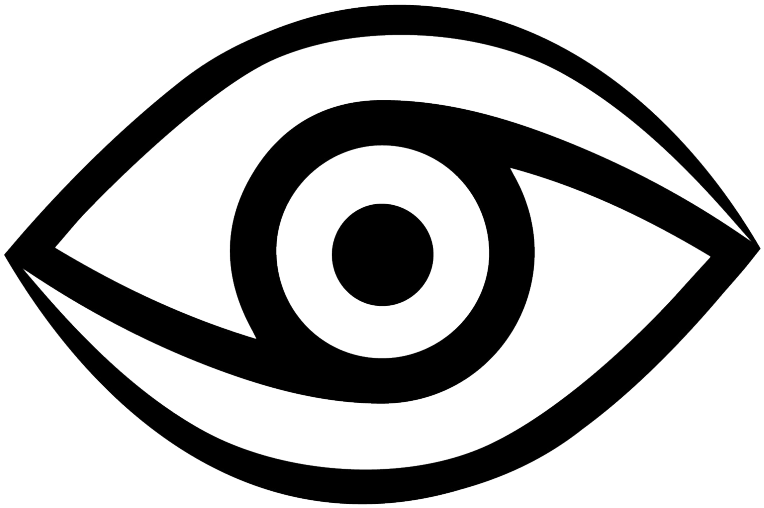
Project Plan

TradeX



Date: 28.02.2024
Version: 1.0
Author: Solovenko Danila

Project assignment 4
Context 4
Goal of the Project 4
Scope and precoditions 4
Strategy 6
Research questions and methodology 6
End products 7
Project organisation 8
Stakeholders and team members 8
Communication 9
Activities and time plan 9
Phases of the project 9
Time plan and milestones 11

Project assignment

Context

TradeX is a project aimed at developing a trading tool for strategy visualization. The project is initiated and managed independently as part of a university assignment or personal endeavor.

Goal of the Project

The primary objective of the TradeX project is twofold:

Why this Project:

1. *Integration with Side Project*: The main motivation behind developing TradeX is to seamlessly connect it with the existing side project. By doing so, the initiator aims to enhance the data analysis capabilities and provide valuable insights into market trends and trading signals.
2. *Comprehensive Visualization*: TradeX is designed to offer advanced visualization features tailored specifically for analyzing market charts and identifying trading signals. This serves as a critical component in the decision-making process for traders and investors.

Scope and precoditions

Indside scope:

1. Prebuild strategy visualization.
2. Custom strategy building tool with python.
3. Admin panel with two types of users.
4. Profit calculator for a given time period.
5. Documetation update every sprint.
6. Backlog and retrospective every sprint.

Outside scope:

1. Any order placement logic.
2. Automatic trading inside the app.

In moving forward with the TradeX project and aligning it with the side project, certain preconditions and technology choices have been made. Here are the key considerations:

1. Java for Backend:
    - Rationale: Java is chosen for its robustness, scalability, and extensive ecosystem of libraries and frameworks suitable for building backend systems.
    - Critical Analysis: Java offers excellent performance and reliability, but it's essential to ensure that the chosen frameworks and libraries align with the project's requirements and scalability needs.
2. React for Frontend:
    - Rationale: React is selected for its component-based architecture, virtual DOM, and strong community support, making it ideal for building interactive and dynamic user interfaces.
    - Critical Analysis: While React is widely adopted and offers numerous benefits, it's crucial to consider factors such as learning curve, state management, and performance optimization during development.

Tinkoff API for Market Data:

- Rationale: Tinkoff API provides access to both historical and live market data, offering a comprehensive source for analyzing market trends and trading signals.
- Critical Analysis: While Tinkoff API may offer the required functionality, it's essential to evaluate factors such as data reliability, latency, and API rate limits to ensure seamless integration with the trading tool.

Docker for Microservices:

- Rationale: Docker enables containerization, facilitating the deployment and scaling of microservices-based architectures, which can enhance flexibility, scalability, and maintainability.
- Critical Analysis: While Docker simplifies deployment and management, it's important to carefully design the microservices architecture, considering factors such as service boundaries, communication protocols, and orchestration tools for optimal performance and resilience.

Strategy

Agile with scrum justicafication:

1. Flexibility and Adaptability:
   - Agile methodologies, such as Scrum, emphasize flexibility and adaptability to changing requirements and priorities. This aligns well with the dynamic nature of software development projects, allowing for iterative development and continuous improvement.
2. Incremental Delivery:
   - Scrum enables incremental delivery of features and functionalities in short, time-boxed iterations known as sprints. This approach allows for early and frequent delivery of value to stakeholders, fostering feedback and validation throughout the development process.
3. Risk Mitigation:
   - By breaking down the project into smaller, manageable tasks and delivering increments of functionality regularly, Scrum helps mitigate risks associated with large-scale development efforts. Early identification of issues and adaptation based on feedback minimize the likelihood of costly errors or deviations from project objectives.
4. Continuous Improvement:
   - Scrum promotes a culture of continuous improvement through retrospective meetings at the end of each sprint. Team members reflect on what went well, what could be improved, and action items for enhancing productivity and effectiveness in subsequent iterations.

Research questions and methodology

RQ1: What are the most effective visualization techniques for presenting trading strategies and market data?

- Approach/Methodology:
  - Dot Framework:
    - Design: Explore various visualization techniques, such ascandlestick charts, line charts, and heatmaps, through literature review and experimentation.
    - Operationalization: Implement prototypes of different visualization techniques within the project environment.
    - Testing: Gather feedback from stakeholders through user testing and surveys to evaluate the effectiveness and usability of each technique.

        - Analysis: Analyze the collected data to identify the most effective visualization techniques based on criteria such as clarity, interpretability, and user preference.

RQ2: What are the optimal strategies for integrating a Python editor into the web platform for custom strategy visualization and implementing a money calculator for a specific period?

- Approach/Methodology:
    - Dot Framework:
        - Design: Research available options for embedding a Python editor into web applications and identify best practices for integrating such editors seamlessly.
        - Operationalization: Develop prototypes to test different integration approaches, considering factors such as performance, security, and ease of use.
        - Testing: Conduct usability testing and performance testing to evaluate the effectiveness and user experience of the integrated Python editor within the web platform.
        - Analysis: Assess the results to determine the optimal strategy for integrating the Python editor, as well as for implementing a money calculator feature, taking into account user feedback and project requirements.

End products

1. **Project Plan**:
    - Description: Comprehensive document outlining the project scope, objectives, deliverables, schedule, and resource allocation. It serves as a roadmap for project execution and management.
2. **Requirements Document**:
    - Description: Document detailing the functional and non-functional requirements of the TradeX trading tool. It specifies the features, functionalities, and constraints that the software must satisfy.
3. **Architecture Document**:
    - Description: Detailed document describing the overall architecture of the TradeX trading tool, including system components, modules, interfaces, and data flow. It provides a blueprint for software development and maintenance.
4. **Research Reports**:

- Description: Reports summarizing the findings and insights from research activities conducted during the project, such as market analysis, user studies, and technology evaluations. They provide valuable information for decision-making and future reference.

5. **Design Documents**:
   - Description: Documents detailing the design specifications and rationale for the TradeX trading tool, including user interface designs, database schema, and system workflows. They guide the implementation phase and ensure consistency across development efforts.
6. **Test Reports**:
   - Description: Reports documenting the results of testing activities performed on the TradeX trading tool, including unit tests, integration tests, and user acceptance tests. They verify the correctness, reliability, and performance of the software.
7. **Implementation Codebase**:
   - Description: Source code repository containing the implementation of the TradeX trading tool in Java for the backend, React for the frontend, and any additional technologies or libraries used.

Project organisation

Stakeholders and team members

- Stakeholders:
  1. Danila (Project Initiator):
     - Role: Project Sponsor/Owner
     - Description: Responsible for initiating and overseeing the TradeX project, defining its objectives, and ensuring its successful completion. Also involved as a team member in the development process.
- Team Members:
  1. Danila (Project Initiator):

- Role: Developer
- Description: Actively involved in the development of the TradeX trading tool, contributing to coding, testing, and other software development activities.

Communication

- Retrospective:
    - Sprint Project retrospective (small):
        - Goal: Review progress, think of any challenges or blockers, and plan upcoming tasks.
        - Frequency: sprint to sprint

Activities and time plan

Phases of the project

Initiation Phase:

Problem Analysis:

- Define project objectives, scope, and requirements.
- Conduct market research and analysis to identify target users and competition.
- Determine project constraints, such as time, budget, and resources.

Planning Phase:

Project Planning:

- Develop a project plan outlining tasks, deliverables, timelines, and resource allocation.
- Define roles and responsibilities for team members and stakeholders.
- Establish communication channels and meetings schedule.

Development Phase:

Software Development:

- Design and implement the TradeX trading tool according to the defined requirements and specifications.
- Conduct iterative development sprints using Agile methodologies like Scrum.
- Regularly review progress, address issues, and adapt to changing requirements.

Testing and Quality Assurance Phase:

Quality Assurance:

- Perform comprehensive testing, including unit tests, integration tests, and user acceptance tests.
- Ensure software functionality, reliability, security, and performance meet defined criteria.
- Address and resolve any identified issues or bugs.

Deployment Phase:

Deployment and Launch:

- Prepare deployment artifacts and documentation.
- Deploy the TradeX trading tool to production environments.
- Monitor and validate system performance and stability post-deployment.

Reflection and Wrap-Up Phase:

Reflection:

- Reflect on the project journey, successes, challenges, and lessons learned.
- Capture insights and experiences for personal and professional growth.
- Identify opportunities for future projects or improvements.

Wrap-Up:

- Archive project documentation, code repositories, and other project artifacts.
- Complete administrative tasks, such as final reports, presentations, and project closure documentation.
- Celebrate achievements and acknowledge contributions from team members and stakeholders.

Time plan and milestones

Sprint Length and Justification:

- Sprint Length: 3 weeks
    - Justification: The 3-week sprint length aligns with the requirements of the university and provides an appropriate balance between development time and iteration frequency. It allows for significant progress to be made within each sprint while also accommodating comprehensive testing and feedback cycles.

Organization of Agile Events:

- Sprint Planning:
    - Frequency: Once at the beginning of each sprint.
    - Objective: Define the sprint goal and select user stories for implementation.
- Retrospective:
    - Frequency: At the end of each sprint.
    - Objective: Reflect on the sprint, identify successes and areas for improvement, and plan action items for the next sprint.

Artefacts Planned for Agile Project:

- Product Backlog:
    - Maintained throughout the project, containing a prioritized list of user stories and features.
- Sprint Backlog:
    - Created at the beginning of each sprint, detailing the user stories and tasks committed to for the sprint.

Testing Strategy and configuration management

Testing strategy

1. Unit Testing:
    - Goal: Achieve high code coverage and validate the correctness of individual units of code, such as functions and methods.
2. Component Testing:

- Goal: Validate the behavior and interactions of components within the application.

3. Integration Testing:
    - Goal: Verify the integration and interaction between different modules or services within the system.
4. System Testing:
    - Goal: Validate the functionality and performance of the entire system as a whole.
5. Acceptance Testing:
    - Goal: Ensure that the system meets the business requirements and user expectations.

Quality Testing Setups:

- Code Quality Analysis: Utilize tools like SonarQube to perform static code analysis, detect code smells, and ensure adherence to coding standards.
- Performance Testing: Conduct performance testing using tools like JMeter or Gatling to evaluate the system's scalability, responsiveness, and stability under various load conditions.
- Security Testing: Perform security testing using tools like SonarQube Security Plugin to identify and mitigate potential security vulnerabilities in the application code.

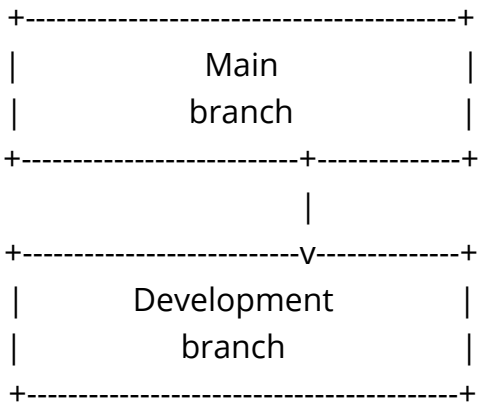
Configuration management

CI/CD workflow:

- Continuous Integration (CI):
    - I regularly integrate my code changes into a shared GutHub repository.
    - Automated build and test processes are triggered whenever new code changes are pushed to the repository (main branch).
    - CI tool «GitHub actions» are used to orchestrate the CI pipeline.
- Continuous Deployment (CD):

- After passing automated tests in the CI pipeline, code changes are deployed to deployment workflow.

Branch diagram:



Risks

Risk and mitigation

Technical Complexity	-Break down project into smaller, manageable tasks. -Conduct regular technical reviews. - Prioritize tasks based on complexity.	-Allocate additional time and resources for resolving challenges. -Seek external expertise if needed.
Resource Constraints	-Conduct resource planning and allocation early in project lifecycle. -Monitor resource utilization. -Explore alternative resource options.	-Reallocate internal resources if possible. -Seek additional funding or outsourcing options.
Security Vulnerabilities	-Implement security best practices throughout development lifecycle. -Regular security assessments -Encryption of sensitive data.	-Develop response plan for addressing security incidents. -Implement incident detection and containment procedures.