Software Project Management Plan (SPMP)

Stock Trends Analysis Dashboard (STAD)

1 Introduction

This document provides the Software Project Management Plan (SPMP) for the Stock Trends Analysis Dashboard (STAD) project, which aims to create a visualization tool for stock data. This SPMP supplements the previously submitted Project Proposal and Software Configuration Management Plan (SCMP) by focusing on project organization, management tools, risk management, estimate, scheduling, and progress monitoring. The goal of this document is to explain how the project will be organized, planned, managed, and monitored throughout its life cycle.

1.1 Overview

The goal of this project is to develop an interactive Stock Trends Analysis Dashboard (STAD) that takes stock market data as input and outputs an easy-to-understand dashboard with data analysis, strategy backtesting, and machine-learning-based trend classification. The system includes interactive visualizations, technical indicators, and historical performance evaluation to support financial exploration. The project involves creating a frontend dashboard, backend data processing, and integration of baseline machine learning models for trend classification.

1.2 Project Deliverables

The final iteration of the project repository shall be submitted by 04/28/2026 and shall comprise the project source code, tests, and documentation. In addition to this, several documents will need to be delivered at various stages of the project. They are as follows:

- Project Proposal 02/03/2026
- Software Configuration Management Plan 02/10/2026

- Software Project Management Plan 02/17/2026
- Software Requirements Specification 02/24/2026
- Mid-semester Presentation 03/03/2026
- Software Design Document 03/24/2026
- Final Presentation 04/28/2026

The final iteration of the project repository shall comprise the project source code, tests, and documentation.

The frontend dashboard source code is maintained in the following repository: `https://github.com/kuanysh100322/stock-trends-analysis-dashboard-frontend`

The backend application source code is maintained in the following repository: `https://github.com/atharvaDS05/Backend-Structure`

1.3 Evolution of the Software Project Management Plan

Version	**Primary Author(s)**	**Description**	**Date**
Draft	Project Manager	Initial draft prepared based on course lectures, project proposal, and SCMP.	02/10/2026
Preliminary	Project Team	Updated version based on team feedback and discussion.	02/16/2026
Final	Project Manager, QA/Documentation Support	Final version submitted for assignment and maintained under change control.	02/17/2026

1.4 Reference Materials

- IEEE Std 1058-1998, *IEEE Standard for Software Project Management Plans.*
- ANSI/IEEE Std 1042-1987 (R1993), *IEEE Guide to Software Configuration Management.*

1.5 Definitions and Acronyms

SCMP	Software Configuration Management Plan
SPMP	Software Project Management Plan
SRS	Software Requirements Specification
SDD	Software Design Document
STAD	Stock Trend Analysis Dashboard

2 Organization

This section describes the process model, organizational structure, and team responsibilities for the Stock Trends Analysis Dashboard (STAD) project.

2.1 Process Model

The STAD project follows an iterative and incremental software process model, as outlined in the Module 2 lecture. Project activities, such as planning, requirements analysis, software design, implementation, and testing, are completed in stages during the semester. The project is divided into several stages, with major deliverable documents submitted consecutively for review and evaluation. The instructor grades the evaluated document and provides necessary feedback. While the feedback from these reviews is integrated into future project artifacts. This technique allows for controlled system evolution while guaranteeing alignment with stated milestones and course requirements.

2.2 Organizational Structure

The STAD project is conducted as part of the CS673 Software Engineering course at BU MET. The project follows a self-contained, project-oriented team structure that is consistent with no external organizational dependencies. The team is structured as a small, cross-functional software team, with roles assigned depending on experience in frontend development, backend systems, machine learning, quality assurance, and project management. This structure promotes effective communication, task ownership, and collaboration throughout the project's lifecycle.

2.3 Project Responsibilities

The STAD project has five key roles. The final four roles report to the Project Manager, who is in charge of overall project coordination and delivery.

2.3.1 Project Manager

The project manager is responsible for planning and coordinating overall project activities, managing the project schedule, organizing team meetings, tracking the completion of project tasks, documenting project status, and ensuring that deliverables are completed on time.

2.3.2 Dashboard/UI Developer

The Dashboard/UI Developer is responsible for design and implementation of the dashboard user interface using React and Vite; definition of page structure and navigation; development of reusable UI components, and integration of frontend components with backend REST APIs.

2.3.3 Data/Backend Developer

The Data / Backend Developer is responsible for design and implementation of backend services and APIs, supporting database and data processing pipelines, ensuring robust system integration between frontend and backend components.

2.3.4 ML/QA Engineer

The ML / QA Engineer is responsible for design, implementation of baseline machine learning models, evaluation of model performance, and contribution to testing and quality assurance activities.

2.3.5 QA and Documentation Support Engineer

The QA and documentation support engineer is in charge of contributing to the project's three primary areas (dashboard/UI, data/backend, and ML), as well as ensuring their interaction. This role includes reviewing documentation for clarity and consistency, supporting testing activities, and assisting with frontend-backend integration and validation.

3 Managerial Process

3.1 Management Objectives and Priorities

The primary management goal of the STAD project is to effectively provide all necessary project artifacts and system components in accordance with the CS673 course schedule and specifications. This includes completing and submitting documents on time, implementing fundamental system functionality, and preparing course presentations.

3.2 Assumptions, Dependencies, and Constraints

The project plan is based on several assumptions. It is assumed that all team members will actively participate throughout the semester and complete assigned tasks according to the agreed schedule. It is also assumed that all required technologies (React, Django, Python ML libraries) will function as expected without major compatibility issues. The project depends on the successful integration between frontend, backend, and machine learning components. Timely completion of the SRS and SDD documents is necessary before full implementation begins. The project also depends on GitHub availability for version control and collaboration. The primary constraint of this project is time. The schedule is strictly defined by course deadlines. There is no financial budget, and all tools used must be open-source or freely available. The scope of the project must remain manageable to ensure timely delivery and sufficient testing before final submission.

3.3 Risk Management

This project includes a lightweight machine learning (ML) component and a data-driven analytics pipeline. The primary risks involve data reliability, reproducibility, integration stability, and ensuring that ML results are interpretable and not overstated. Risks are managed through early validation, controlled project scope, and clearly defined acceptance criteria for each milestone.

Risk 1: Market Data Quality and Schema Issues

Description: Market data obtained from Yahoo Finance may contain missing dates, split or dividend effects, inconsistent column formats, or unexpected null values. These issues can disrupt indicator computation, labeling rules, and feature generation.

Impact: High (pipeline failure or misleading outputs).

Likelihood: Medium.

Mitigation: Implement strict input validation and normalization in the backend, including required column checks, date sorting, and numeric validation. Use adjusted close prices where appropriate and document all data assumptions.

Owner: Backend / ML Owner.

Risk 2: Feature Leakage and Unrealistic ML Evaluation

Description: Time series data can unintentionally include future information in features or use random train/test splits, resulting in inflated performance metrics.

Impact: High (invalid ML results).

Likelihood: Medium.

Mitigation: Use time-aware data splits (training on earlier windows and testing on later windows). Clearly define prediction horizons and prevent future data from being used during feature generation. Document evaluation methodology and metrics in the repository.

Owner: ML / QA Owner.

Risk 3: ML Scope Creep

Description: Expanding ML functionality beyond baseline models (e.g., extensive hyper-parameter tuning or advanced forecasting) may exceed project scope and available time.

Impact: Medium–High (missed deadlines or unstable system).

Likelihood: High.

Mitigation: Limit ML implementation to one or two baseline models (e.g., Logistic Regression or Random Forest) with a fixed feature set and evaluation plan. Treat ML as a supporting component rather than a core dependency of the dashboard.

Owner: Project Manager and ML Owner.

Risk 4: Reproducibility Differences Across Environments

Description: ML results may vary due to differences in library versions, operating systems, or random seeds.

Impact: Medium (inconsistent results).

Likelihood: Medium.

Mitigation: Pin dependency versions in `requirements.txt`, set random seeds explicitly, and record execution parameters such as ticker, date range, interval, model type, and seed.

Owner: Backend / ML Owner.

Risk 5: Performance and Rate Limiting

Description: Repeated on-demand data retrieval and indicator computation may degrade dashboard performance or trigger rate limits from external services.

Impact: Medium (slow response times or timeouts).

Likelihood: Medium.

Mitigation: Cache results for recently requested tickers and date ranges with a short time-to-live (TTL). Apply request throttling and input limits, and compute indicators once per execution rather than per visualization request.

Owner: Backend Owner.

Risk 6: QA Coverage Gaps and Incorrect Analytics

Description: Minor implementation errors in indicators, labeling logic, or backtesting can produce plausible but incorrect outputs.

Impact: High (incorrect analytics and reduced credibility).

Likelihood: Medium.

Mitigation: Implement unit tests for indicator calculations using known datasets, integration tests for end-to-end flows, and sanity checks on outputs (e.g., return ranges and non-negative volumes). Define acceptance criteria for each major feature.

Owner: QA and ML Owner.

3.4 Monitoring and Controlling Mechanisms

Project monitoring and control are carried out by structured team coordination and regular communication practices, as highlighted in the Project Team Guidance of Module 1 lecture. An initial team meeting was conducted through a dedicated Discord channel provided for the project group 4. This channel was used to discuss a research topic, establish meeting schedules, team members' skills and areas of expertise for assigning appropriate roles and responsibilities. Project progress is tracked through weekly meetings in which team members provide status updates, report completed work, and identify further tasks. Google Docs is used as a collaborative workplace for creating and collectively reviewing project documentation. GitHub serves as the primary platform for managing source code and version control, as well as for maintaining completed project artifacts, including documentation. Together, these tools enable effective monitoring of project timeline, quality, and functionality, as well as the regulated evolution of project artifacts throughout the project lifecycle.

4 Technical Process

4.1 Methods, Tools, and Techniques

Front-end development is conducted using React and Vite following a modular, component-based architecture. Development tasks are divided into manageable units aligned with dashboard features, including data upload, visualization rendering, and indicator configuration. GitHub is used for version control, and changes are committed frequently with descriptive commit messages. Integration with backend APIs is tested incrementally to ensure stability.

Development follows an iterative approach, with incremental feature implementation and regular testing before integration into the main branch.

4.2 Software Documentation

The software documentation list for this project includes the following documents:

- Project Proposal
- Software Configuration Management Plan (SCMP)
- Software Project Management Plan (SPMP)
- Software Requirements Specification (SRS)
- Software Design Document (SDD)

Section 1.2 (Project Deliverables) specifies the milestones and submission dates for these documents.

4.3 Project Support Functions

To ensure the Stock Trends Analysis Dashboard is reliable, reproducible, and easy to integrate, the team applies a layered support strategy covering data validation, testing, and evaluation. These support functions are designed to protect the end-to-end user workflow (select ticker → generate analytics → display dashboard) and to ensure that results presented to users are consistent and defensible.

Data Validation and Pre-Processing. Since downstream computations depend on clean time-series data, the backend validates inputs immediately after retrieval. Validation includes schema and type checks (required OHLCV fields, numeric parsing), date sorting and duplicate detection, missing value handling, and basic sanity checks such as non-negative volume and realistic price fields. Warnings are surfaced when data quality may affect indicators, labels, or backtesting results.

Testing Strategy. The project uses a combination of unit, API, and integration testing. Unit tests focus on correctness of indicator calculations, labeling rules, and backtesting outputs using known datasets. API tests verify endpoint behavior, status codes, and response structure stability. Integration tests validate the full pipeline (retrieve data → compute indicators → apply labels → run backtest → return results). Tests are executed prior to merging changes into the main branch.

Model Evaluation and Result Verification. If optional ML evaluation is implemented, it is treated as an educational comparison layer rather than a production trading system. Models are evaluated using time-aware splits to avoid data leakage. Evaluation outputs include standard classification metrics and record run configuration details (ticker, date range, interval, model type, random seed). Non-ML outputs are verified using sanity checks such as indicator range validation and comparison against buy-and-hold strategies.

Operational Support and Documentation. Concise documentation is maintained for local setup, API usage, and expected inputs and outputs. All project artifacts (code, tests, SCMP, SPMP, and supporting documentation) are maintained in GitHub repositories. Logging is included for key pipeline steps to support debugging and demonstrations.

5 Work Elements, Schedule, and Budget

5.1 Work Packages

The tasks required for the STAD project are divided into the following work packages.

5.1.1 Backend and Data Handling

The backend and data handling are the primary responsibility of the data/backend developer, with support from the QA and documentation support engineer. This package includes:

- API management
- Data handling
- Business logic
- Backend testing
- Backend documentation

5.1.2 Frontend/User Interface

The frontend and user interface are the primary responsibility of the dashboard/UI developer, with support from the QA and documentation support engineer. This package includes:

- Dashboard implementation
- Visualization for machine learning classifier

- Integration with backend services
- Frontend testing
- Frontend documentation

5.1.3 Machine Learning Classifier

The machine learning classifier is the primary responsibility of the ML/QA engineer, with support from the QA and documentation support engineer. This package includes:

- Machine learning classifier implementation
- ML testing
- ML documentation

5.1.4 Process Documentation

Process documentation is the primary responsibility of the project manager, with support from all team members. This package includes:

- Project Proposal
- Software Configuration Management Plan
- Software Project Management Plan
- Software Requirements Specification
- Mid-semester Presentation
- Software Design Document
- Final Presentation

5.2 Dependencies

The dependencies among the work packages are minimal, with the following exceptions:

- The data format must be defined before the dashboard or machine learning components can be tested.
- The machine learning classifier must be functional before ML visualizations can be implemented.

- A working dashboard prototype is required for the mid-semester presentation, and a final system is required for the final presentation.

5.3 Resource Requirements

The primary resources for this project are human resources. Each team member requires access to a personal computer. Meetings are conducted either in person at the Kenmore Classroom Building at Boston University or virtually via Discord. No additional physical equipment is required, as GitHub-hosted repositories provide sufficient infrastructure for the project.

5.4 Schedule

The schedule for final project deliverables is defined in Section 1.2. The planned development schedule for the codebase is as follows:

Working prototype (for presentation)	03/01/2026
All code finalized and tested	04/14/2026
All code documented	04/21/2026

This schedule allows a two-week buffer for development and testing delays and a one-week buffer for the overall project timeline.

6 Team Member Contributions

Ulzhalgas Seidaliyeva – Project Manager. Ulzhalgas Seidaliyeva coordinated the development of the SPMP following the SCMP submission. Responsibilities included organizing the document according to IEEE guidelines, incorporating feedback from prior submissions, coordinating task assignments, and ensuring consistency across project artifacts.

Kuanysh Amandos – Dashboard/UI Developer and Data/Backend Developer. Since the SCMP submission, Kuanysh Amandos designed the frontend technology stack using React and Vite and defined the UI component architecture. He outlined frontend responsibilities, including data upload, visualization rendering, indicator configuration, and incremental API integration using GitHub-based version control.

Haoqian Zhang – Dashboard/UI Developer and Data/Backend Developer. Since the SCMP submission, Haoqian Zhang identified backend work packages, defined API endpoints and data dependencies, and documented the data processing pipeline and integration points between backend and frontend components.

Atharva Sharma – ML/QA Engineer. Since the SCMP submission, Atharva Sharma defined the ML scope and risks, contributed to ML evaluation planning, and supported quality assurance activities, including testing strategies and validation methods for ML components.

Mackenzie Kong-Sivert – QA and Documentation Support. Mackenzie Kong-Sivert developed the overall document outline and completed previously missing sections, including substantial contributions to Section 5.