Software Requirements Specification for the "NSUTS for VS Code" Project

1. Authors

- Anna Vladimirovna Shemchuk
- Stepan Vladimirovich Efimov
- Alina Aleksandrovna Maslova

2. Introduction

This document describes the requirements for the "NSUTS for VS Code" extension for Visual Studio Code. The extension is designed to integrate the NSUTS testing system into the VS Code development environment, allowing participants in programming competitions to solve problems without switching between browser and code editor windows. The main goal is to significantly speed up and simplify the process of submitting solutions and receiving results, saving critical time during competitions.

3. Glossary

* NSUTS: A system designed for conducting programming competitions and verifying participants' solutions.

* Contest: An NSUTS event in which participants solve tasks.

* Round: A stage of a competition that groups a set of tasks.

* Task: A specific task to be solved, defined by a condition, constraints, and a set of tests.

* Submission Report: The result of the solution review, including status (OK, Compilation Error, Incorrect Answer, etc.) and test details.

* Leaderboard: A leaderboard displaying the current results of contest participants.

4. Actors

Participant

Definition: A VS Code user participating in an NSUTS olympiad or contest.

Goals: Efficiently solve tasks, submit solutions, track results and rankings without leaving the development environment.

Responsibilities: Log in to the system, select a contest and problem, write code, submit solutions, view results and the leaderboard.

5. Functional Requirements

5.1. Strategic Use Cases

5.1.1. Use Case `UC-S-1`: Participate in a Contest

5.1.2. Use Case `UC-S-2`: View Progress and History

5.2. Use Cases for Contestant

5.2.1. Use Case `UC-1-1`: Authentication in the System

Actors: Contestant

Goals: The contestant wants to access the plugin's functionality using their NSUTS credentials.

Precondition: The "NSUTS for VS Code" extension is installed. The user has a valid NSUTS account.

Trigger Condition: The user runs the "NSUTS: Login" command or attempts to access a protected plugin function.

Main Success Scenario:

1) The system displays a dialog box with login and password fields.

2) The contestant enters their credentials and confirms their entry.

3) The system sends the data to the NSUTS server for verification.

4) The system receives and stores the authentication token.

5) The system notifies the contestant of successful login and opens the main control panel. Alternative scenario 'Invalid credentials':

Trigger condition: In step 3, the NSUTS server returns an authentication error.

1) The system displays the error message "Invalid login or password."

2) The scenario returns to step 1 of the main scenario.

5.2.2. Use case `UC-1-2`: Viewing and selecting a contest/task

Actors: Participant

Goals: The participant wants to see a list of available contests and tasks and select a problem to solve.

Precondition: The participant is successfully authenticated (`UC-1-1`).

Trigger condition: The user opens the "Contests" tab in the plugin panel.

Main success scenario:

1) The system requests and receives a list of active contests from the NSUTS server.

2) The system displays the list of contests in a tree view (Contest -> Tour -> Problem).

3) The participant selects a specific task from the tree.

4) The system displays the condition of the selected task in a separate panel within VS Code.

5.2.3. Use-case `UC-1-3`: Submitting a solution for review

Actors: Participant

Goals: The participant wants to submit the written code for review in NSUTS.

Precondition: The participant is authenticated. A file (or several files) with the solution is open. The task is selected in the plugin panel.

Trigger condition: The user clicks the "Submit" button in the plugin panel or runs the "NSUTS: Submit Solution" command.

Main success scenario:

1) The system identifies the currently active file as the solution.

2) The system provides an interface for selecting a compiler (programming language) from the list available for this task.

3) The participant confirms the compiler selection and submission.

4) The system packages the solution file(s) and sends them to the NSUTS server along with the task ID and the selected compiler.

5) The system displays the notification "Solution submitted for review" and begins tracking the review status.

Alternative scenario `Submitting multiple files`:

Trigger condition: The participant has previously indicated that the solution consists of multiple files.

1) In step 1, the system packages all files marked as part of the solution into an archive.

2) The main scenario follows, starting with step 2.

5.2.4. Use-case `UC-1-4`: Viewing an attempt report

Actors: Participant

Goals: The participant wants to see the detailed review result of their submission (status, passed tests, compilation errors, etc.).

Precondition: At least one submission attempt has been made (`UC-1-3`).

Trigger condition: The system automatically receives a submission status update, or the user manually updates the attempt history.

Main success scenario:

1) The system requests the status of the latest attempts from the NSUTS server.

2) The system displays a list of attempts in a dedicated panel with a brief status for each (OK, WA, CE, etc.).

3) The participant selects a specific attempt from the list.

4) The system displays a detailed report: the status of each test, time and memory, and compilation error text, if applicable.

5.2.5. Use case `UC-1-5`: Viewing the Leaderboard

Actors: Participant

Goals: The participant wants to see the status of the contest (their own ranking and the rankings of other participants).

Precondition: The participant is authenticated and participating in the contest.

Trigger condition: The user opens the "Leaderboard" tab in the plugin panel. Main success scenario:

1) The system requests up-to-date leaderboard data from the NSUTS server.

2) The system displays a table with rankings, participant names, number of solved problems, and penalty time.

3) The system periodically (or by pressing the refresh button) repeats the request to ensure the data remains current.

5.2.6. Use case `UC-1-6`: Viewing attempt history

Actors: Participant

Goals: The participant wants to view all their previous submissions for all tasks, with filtering options.

Precondition: The participant is authenticated.

Trigger condition: The user opens the "Attempt History" tab.

Main success scenario:

1) The system requests the user's full attempt history.

2) The system displays a table with the date, task, attempt status, and selected compiler.

3) The participant can filter the history by task, contest, or status.

5.2.7. Use case `UC-1-7`: Interacting with organizers

Actors: Participant

Goals: The participant wants to ask the organizers a question about the task or get clarification.

Precondition: The participant is authenticated and participating in the contest.

Trigger condition: The user clicks the "Ask a Question" button in the task viewing interface.

Main success scenario:

1) The system displays a dialog box for entering the question.

2) The participant enters the question and submits it.

3) The system sends the question to the NSUTS server, linking it to the current problem.

4) The system displays the history of submitted questions and answers from the organizers in a separate panel.

5.2.8. Use case `UC-1-8`: Viewing FAQ

Actors: Participant

Goals: The participant wants to get an answer to a frequently asked question without contacting the organizers.

Precondition: The participant is authenticated.

Trigger condition: The user opens the "FAQ" tab.

Main success scenario:

1) The system downloads the current list of questions and answers from the NSUTS server.

2) The system displays the FAQ in a readable format.

6. System-wide functional requirements

- Security
- Networking (Asynchronous requests without blocking the interface)
- Integration with VS Code
- Reliability (Preserving state between restarts)

7. Non-functional requirements

7.1. Environment

- Support for VS Code version 1.104.0 and higher
- Compatibility with Windows 10+, macOS 12+, Ubuntu 20.04+
- A stable internet connection is required
- Compatibility with the NSUTS API

7.2. Performance

- Fast loading of the contest list and solution submission
- The interface does not block during network requests

7.3. Reliability

- Automatic session recovery in case of connection loss
- Data preservation in case of VS Code crash
- Notifications about NSUTS server unavailability

7.4. Extensibility

- Modular architecture for adding new features
- Configurable via configuration file
- API for internal computers