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

1. Authors

- Anna Vladimirovna Nerlikh
- 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 of programming competitions to solve tasks without switching between browser and code editor window. The main goal is to significantly speed up and simplify the process of sending solutions and receiving pass report, 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.

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

4. Actors

Participant

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

Goals: Efficiently solve tasks, send solutions, get pass report 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.

Main Success Scenario:

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

2) The participant enters their credentials and confirms the input.

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

4) The system receives and stores the authentication cookie.

5) The system notifies the contestant of successful login and opens the main control panel.

Alternative scenario 'Invalid credentials':

- 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 -> Task).

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

4) The system parses the round's common PDF file, extracts the condition of the selected task from it, and displays the task text and examples in a separate panel within VS Code.

5.2.3. Use-case UC-1-3: Sending a solution for review and viewing the pass report

Actors: Participant

Goals: The participant wants to submit the written code for review in NSUTS and view the pass report.

Precondition: The participant is authenticated. A file (or multiple files are selected) 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 provides an interface for selecting files.
2. The system provides an interface for selecting a compiler from the list available for this task.
3. The system packages the solution file(s) and sends them to the NSUTS server along with the task ID and the selected compiler.
4. The system displays the notification "Solution submitted for review" and begins tracking the review status.
5. The system requests the status of the latest attempts from the NSUTS server.
6. The system displays the last attempt in a dedicated panel with a brief status (OK, WA, CE, etc.).

Alternative scenario:

Submitting multiple files: 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 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.

3) The ability to filter history by task, contest, or status.

5.2.5. Use case `UC-1-5`: Log out

Actors: Participant

Goals: Log out of the system.

Precondition: The participant is authenticated.

Trigger condition: the user click the “Exit” button.

Main success scenario:

1) The system terminates the current user session.
2) The system clears the stored authentication cookie.
3) The system updates the plugin interface, hiding elements available only to authenticated users and showing the login prompt.

6. System-wide functional requirements

Our system supports not just isolated actions, but the entire contest workflow – right inside VS Code. A participant writes code, submits a solution, and sees the result immediately, without switching to a browser.

But what if they accidentally submit the wrong file or lose track of which task they’re working on?

To prevent this, the extension provides a unified task context, reliable session persistence, and real-time result updates. It always knows exactly which contest and which task you’re working on – and ensures every action is tied to that context. You log in once, and everything works until you choose to log out yourself. This isn’t just convenient - it’s a system-wide requirement that saves time and prevents mistakes under contest conditions.

7. Non-functional requirements

7.1. Environment

• VS Code version: the extension is tested and fully compatible with VS Code 1.104.0 and higher.

• A stable connection with a minimum download/upload speed of 1 Mbps is required for reliable communication with the NSUTS server

• The extension is designed to work exclusively with the official NSUTS API; any changes to the API must be reflected in the extension to maintain functionality.

7.2. Performance

• The list of active contests and tasks must load within ≤ 2 seconds under normal network conditions (1+ Mbps).

• The submission process (from clicking “Submit” to receiving confirmation of receipt by the server) must complete within ≤ 3 seconds.