TED UNIVERSITY

CMPE491/SENG491 Senior Project

SyntaxSavior Project Specifications Report

25.10.2024

Team Members:

İrem BEŞİROĞLU Software Engineering

Yiğit Oğuzhan KÖKTEN Computer Engineering

Yüksel Çağlar BAYPINAR Computer Engineering

1. Introduction

This document outlines the specifications and requirements for SyntaxSavior, an educational assistance system designed to enhance the learning experience in introductory programming laboratories. The project emerges from observed challenges in current computer science education methodologies, particularly in foundational programming courses where students often encounter difficulties in bridging theoretical knowledge with practical implementation.

The rapid evolution of educational technology, particularly the emergence of Large Language Models (LLMs) and automated coding assistance tools, has created both opportunities and challenges in computer science education. While these tools offer immediate solutions, they may inadvertently hinder the development of fundamental programming skills and problem-solving capabilities. SyntaxSavior aims to address this pedagogical challenge by providing contextual, educational support that guides students through the programming process while maintaining the essential learning experience.

This specification document serves multiple purposes: it establishes the project's scope and objectives, delineates the various constraints (including hardware, software, temporal, financial, environmental, operational, safety, and regulatory considerations), and addresses potential professional and ethical implications. Furthermore, it presents a detailed analysis of system requirements and design considerations necessary for successful implementation within an academic environment.

The following sections provide comprehensive information about:

- Project description and scope
- Technical and operational constraints
- Professional and ethical considerations
- Functional and non-functional requirements
- Implementation guidelines and considerations

Through this documentation, we aim to establish a foundation for developing a system that enhances the educational experience while maintaining academic integrity and promoting genuine learning outcomes in introductory programming courses.

Description

SyntaxSavior is an innovative educational tool designed to transform the introductory programming laboratory experience by providing real-time, contextual assistance to students while preserving the essential learning process. The system consists of two main components: an IDE plugin and a main analysis program working in concert to create an interactive learning environment.

1.1.1. Core Functionality

The system's primary function is to serve as an intelligent learning assistant that bridges the gap between theoretical knowledge and practical programming experience. This is achieved through:

Real-time Error Analysis and Guidance

- The IDE plugin continuously monitors student code during laboratory sessions
- Syntax and compilation errors are immediately detected and transmitted to the main program
- The main program processes these errors and provides detailed, step-by-step explanations in student-friendly language
- Complex error messages are translated into comprehensible instructions that relate to course concepts

Contextual Learning Support

- The system maintains awareness of the current laboratory task specifications
- Assistance is tailored based on the course materials students have covered
- Guidance is provided within the context of intended learning outcomes
- Solutions are never directly provided; instead, students receive educational hints and explanations

Code Analysis

- Student code is analyzed for logical errors and pattern recognition
- The system identifies potential issues before they lead to runtime errors
- Feedback includes explanations of fundamental programming concepts relevant to the identified issues

1.1.2. Implementation Framework

The project will be initially implemented as an Eclipse plugin, with potential expansion to other IDEs in future iterations. The system will integrate with existing course infrastructure through:

- Direct access to laboratory task specifications
- Integration with course material progression
- Collaboration with course instructors to calibrate assistance levels
- Restricted and controlled interface to prevent over-reliance on automated solutions

1.1.3. Educational Methodology

SyntaxSavior employs a guided discovery approach to learning, where:

- Students are encouraged to understand and solve problems independently
- Assistance is provided through explanation rather than direct solutions
- Core programming concepts are reinforced through practical application
- Real-time feedback creates an interactive learning environment

The system's design prioritizes the development of:

- Problem-solving skills
- Debugging capabilities
- Understanding of programming fundamentals
- Self-reliant learning habits

Constraints

The development and implementation of SyntaxSavior faces several significant constraints that must be carefully considered to ensure its effectiveness as an educational tool. These constraints span technical, operational, and pedagogical domains.

1.1.4. Technical Constraints

Resource efficiency stands as a primary technical constraint, as the system must operate effectively on laboratory computers with limited computational resources. This necessitates careful optimization of the code analysis and feedback mechanisms to ensure smooth operation without impacting the performance of the IDE or the students' coding environment. The integration with Eclipse, while necessary for the current implementation, presents its own challenges given the IDE's complex architecture and the need to maintain a straightforward

user interface that does not add to the existing complexity students face when learning to use development tools.

1.1.5. Educational and Pedagogical Constraints

SyntaxSavior must strictly align with the CMPE113 curriculum, carefully balancing its assistance to neither exceed nor fall short of the course's learning objectives. The system needs to maintain awareness of the current course progression across different class sections, ensuring that guidance and explanations remain relevant to the material students have covered. This requires a dynamic approach to assistance that evolves alongside the course timeline.

A crucial constraint lies in maintaining the delicate balance between assisted and independent learning. The system must provide sufficient support to aid understanding while avoiding over-assistance that might hinder the development of independent critical thinking skills. This constraint becomes particularly challenging given the diverse learning paces and styles present in a typical first-year engineering class, where students bring varying levels of technical competence and prior programming experience.

1.1.6. Operational Constraints

The system must achieve maximum automation in its analysis and guidance capabilities to minimize additional workload for laboratory assistants and tutors. This automation needs to be sophisticated enough to operate without constant supervision while remaining dependable and accurate in its assistance. The constraint of minimizing human intervention must be balanced against the need for accurate and helpful guidance.

1.1.7. Integration and Compatibility Constraints

While initially focused on Eclipse integration, SyntaxSavior must maintain compatibility with enterprise-supported platforms and Learning Management Systems (LMS) used in the academic environment. This integration must be seamless while working within the constraints of existing university infrastructure and security protocols.

1.1.8. Resource and Budget Constraints

The implementation faces budget constraints particularly regarding the use of external APIs and paid tools. This necessitates careful consideration in the selection and integration of language models and other third-party services, potentially requiring the development of alternative solutions that can operate within budgetary limitations.

1.1.9. User Interface Constraints

The interface design faces the significant constraint of needing to be intuitive and accessible to first-year students while operating within Eclipse's existing framework. Given that students are already grappling with Eclipse's relatively complex interface, SyntaxSavior's user interaction elements must be straightforward and unobtrusive, avoiding any additional cognitive load that might impede the learning process.

1.1.10. Security Constraints

The system must implement robust restrictions on LLM usage to prevent misuse or over-reliance on direct solutions. This includes developing sophisticated mechanisms to ensure that assistance remains educational rather than solution-providing, while maintaining security and academic integrity during laboratory sessions.

These constraints form the fundamental framework within which SyntaxSavior must operate, guiding development decisions and implementation strategies to create an effective educational tool that serves its intended purpose while working within established limitations.

Professional and Ethical Issues

The rise of Natural Large Language Models in educational environments presents a critical challenge to academic integrity. The line between educational assistance and potential academic misconduct has become increasingly blurred, with tools like OpenAI's GPT models, Anthropic's Claude, and Google's Gemini becoming prevalent in students' academic workflows. As we develop SyntaxSavior, which will incorporate language model capabilities, it is imperative to implement robust precautions that prevent misuse while preserving the educational value of the system.

In the current educational atmosphere, students have grown accustomed to using these language models as primary tools for handling coursework, often circumventing the intended learning outcomes. However, it is crucial to recognize that the commonly used chat interfaces represent only the most basic form of interaction with these models. SyntaxSavior aims to demonstrate that language models can be implemented in ways that enhance, rather than replace, students' problem-solving skills. This requires careful consideration of our target audience's educational background and their established patterns of AI tool usage.

The implementation of an academically mandated assistance tool raises several sensitive concerns. There is legitimate apprehension about the potential replacement of human

instructors and tutors with digital alternatives. Moreover, we must address the risk of students developing dependency on assistance tools rather than using them as stepping stones toward independent learning. This is particularly crucial given our demographic of 17–20-year-old students, who are at a formative stage in their academic development.

Our data collection and research processes require careful consideration of student privacy rights. While we need to understand student needs and difficulties during lab hours through interviews and usage analysis, this must be done without compromising their personal and digital privacy. The system will operate with complete transparency regarding data handling - analyzing code and errors in real-time without storing or retaining any student data. Students will, however, maintain the option to export their own session data for personal reference.

Security considerations are paramount, given the system's integration with graded laboratory work. The main program will implement measures to restrict access to external AI assistance websites during laboratory sessions, ensuring academic integrity without requiring network-level modifications. This approach maintains a balance between security and accessibility while respecting institutional IT infrastructure.

The system adopts an opt-in approach to various features, allowing students to choose their level of engagement with syntax checking, logic analysis, and step-by-step guidelines. This voluntary participation model ensures that students who prefer traditional learning methods remain unaffected, while those seeking assistance can easily access it. The system's presence should enhance rather than disrupt the laboratory environment.

Regarding the diverse student population in introductory programming courses, we recognize that not all students will pursue computer science as their primary field. While the fundamental goal of developing problem-solving skills remains constant across all engineering disciplines, the system allows for personalized learning experiences. Students can choose the depth and type of assistance they receive, though this customization does not affect the core learning objectives or assessment criteria. This approach maintains educational equity while acknowledging varying career trajectories.

The transparency of system operations presents another ethical consideration. While students should understand that their code is being analyzed, the technical details of this analysis must be balanced against the risk of overwhelming non-technical users. The system will maintain clear communication about its basic functions while avoiding excessive technical detail that might distract from the learning process.

Throughout all these considerations, our primary focus remains on supporting genuine learning outcomes while maintaining academic integrity. The system must serve as a bridge between theoretical knowledge and practical programming skills, without becoming a crutch that hinders independent development.

2. Requirements

The following requirements outline the necessary features and operational capabilities for SyntaxSavior, focusing on functionality, educational support, user interface, performance, security, privacy, and system administration. These requirements ensure that the system is educationally impactful, scalable, and easy to deploy in various introductory programming laboratory settings.

Core Functional Requirements

To support introductory programming courses effectively, SyntaxSavior must provide real-time analysis and educational assistance while integrating seamlessly within the students' coding environment.

- **Real-time Code Analysis Capability**: The system should continuously monitor and analyze student code, detecting errors as they occur.
- **Error Detection and Classification**: SyntaxSavior must classify errors into syntax, runtime, and logical categories, ensuring that students receive specific and actionable feedback.
- **Integration with Eclipse IDE**: The system will integrate with Eclipse as a plugin, allowing easy access and interaction during lab sessions.
- **Step-by-Step Explanation Generation**: For each detected error, SyntaxSavior should generate a step-by-step explanation that relates directly to the course material, promoting understanding over quick fixes.
- **Context-Aware Assistance**: The system should adapt assistance based on the current lab task, allowing for targeted help relevant to the immediate programming objectives.
- **Controlled LLM Integration**: The assistant will interact with language models in a controlled manner, providing educational support while preventing over-reliance on AI-driven solutions.

- **Restriction of External AI Assistance**: SyntaxSavior must be able to limit access to external AI tools during lab sessions to preserve academic integrity.

Educational Requirements

SyntaxSavior is designed to align with CMPE113 curriculum objectives, fostering independent problem-solving skills through a structured, supportive educational framework.

- **Lab Task Specification Integration**: The system must incorporate the specific objectives and requirements of each lab task, ensuring that guidance aligns with the intended learning outcomes.
- **Course Progression Tracking**: SyntaxSavior should track the progression of course material, adapting feedback to align with what students have covered in class.
- **Multi-Level Assistance Options**: The system will offer various levels of help—syntax hints, logical analysis, and task-specific guidance—to support different learning approaches and needs.
- **Clear Distinction Between Hints and Solutions**: SyntaxSavior should ensure that assistance is delivered as hints or explanations rather than direct answers, promoting critical thinking.
- **Support for Diverse Learning Approaches**: Recognizing varied student backgrounds, the system should provide flexible learning pathways, allowing students to select assistance that best fits their learning style.
- **Curriculum Alignment**: The assistance provided should be consistent with CMPE113 learning objectives, focusing on reinforcing foundational programming concepts and debugging skills.

User Interface Requirements

To enhance usability, SyntaxSavior's interface must be clear, intuitive, and non-intrusive, integrating seamlessly within the Eclipse environment.

- **Clear Error Highlighting**: SyntaxSavior will highlight errors directly in the IDE, with distinct visual cues for different types of errors.
- **Feedback Presentation**: Feedback should be presented in easily understandable language, using familiar terminology from the course material.

- **Simple Activation/Deactivation**: The system should feature an easy mechanism for students and instructors to activate or deactivate the plugin as needed.
- **Customizable Assistance Levels**: Users should be able to adjust the level of help they receive, from simple syntax guidance to in-depth error analysis.
- **Non-Intrusive Integration**: The system should integrate with Eclipse in a manner that minimizes disruption to the coding process.
- **System Status Indication**: Clear indicators should inform users of the system's status (active, idle, or error state) and other operational notifications.
- **Easy Navigation**: Students should be able to move between assistance types quickly and easily, without needing to navigate complex menus.

Performance Requirements

For smoother user experience, SyntaxSavior must operate efficiently within lab environments, avoiding latency or excessive resource usage.

- **Quick Response Time**: The system should detect and analyze errors in real time, minimizing delays between user input and feedback.
- **Minimal IDE Impact**: SyntaxSavior must operate without significant performance degradation to Eclipse, ensuring students' work is uninterrupted.
- **Concurrent User Handling**: The system should support multiple concurrent users within a lab environment, maintaining stable operation regardless of class size.
- **Stable Operation**: SyntaxSavior must perform reliably throughout lab sessions, minimizing crashes or unresponsive behavior.
- **Efficient Resource Utilization**: Given the limited computational resources of lab computers, SyntaxSavior should be optimized to function effectively without overtaxing system resources.

Security and Privacy Requirements

To maintain student trust and academic integrity, SyntaxSavior must implement robust security and privacy protections.

- **Secure Code Handling**: The system must ensure that the students' code remains secure, with no risk of unauthorized access or external exposure.
- **No Permanent Data Storage**: SyntaxSavior will operate in real time without retaining student data, unless specifically exported by the user.
- **Optional Session Data Export**: Students should have the option to export session data for personal use, enabling reflection without compromising privacy.
- **Controlled LLM Access**: LLM capabilities should be tightly controlled to prevent misuse for direct solution generation, with safeguards to promote educational support only.
- **Misuse Prevention**: Security mechanisms should be in place to prevent students from circumventing the system for unauthorized solutions, maintaining academic integrity during lab sessions.

System Administration Requirements

SyntaxSavior should be easy to deploy, maintain, and update, reducing the burden on instructors and IT staff.

- **Simple Deployment Process**: The system should require minimal setup, allowing instructors to deploy it in lab environments without extensive technical knowledge or support.
- **Low Maintenance Needs**: SyntaxSavior should be designed for low-maintenance operation, with automatic updates or simple manual update mechanisms as necessary.
- **Basic Monitoring Capabilities**: Instructors and administrators should have access to basic system performance monitoring tools to ensure optimal functionality.
- **Configurable for Various Lab Sections**: SyntaxSavior should be easily adaptable to different lab sections and courses, with customizable options to align with specific curriculum requirements.

3. References

This document consists mostly of our observations during the CMPE 113 Lab hours where we have been tutoring and observing the common student behaviour and problems faced.

TED University CMPE 113 2024 Fall Syllabus

Gocen, A., & Aydemir, F. (2020). Artificial intelligence in education and schools. *Research on Education and Media*, *12*(1), 13–21. https://doi.org/10.2478/rem-2020-0003

Akgun, S., & Greenhow, C. (2021). Artificial intelligence in education: Addressing ethical challenges in K-12 settings. *AI And Ethics*, *2*(3), 431–440. https://doi.org/10.1007/s43681-021-00096-7