Promoting Student Learning by Automatically Managing Resubmission Tokens

Saketh K. Rajesh

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science and Applications

Stephen H. Edwards, Chair

Bob Edmison

Clifford A. Shaffer

August 22, 2025

Blacksburg, Virginia

Keywords: Late Policy, LTI Tool, Automation, Computer Science Education

Promoting Student Learning by Automatically Managing Resubmission Tokens

Saketh K. Rajesh

(ABSTRACT)

In many university computer science courses, strict deadlines can create stress and penalize students who face unexpected challenges. A popular alternative is to give students a few "late passes", "resubmission passes", or "tokens" to use on assignments throughout the semester. While this approach is more flexible and equitable, it creates a significant amount of manual work for instructors, who have to track every request and update deadlines across multiple websites. This thesis tackles that problem in two ways. First, it introduces the EGP Broker, an LMS-integrated LTI 1.3 tool that automates the entire process, removing the need for instructors to manually manage tokens or for students to perform extra steps to use them. Second, this thesis analyzes what happened when a large introductory computer science course switched from a traditional late penalty (a 10% deduction per day) to a manually managed flexible token resubmission based policy. The results indicate that the flexible policy was associated with higher pass rates, improved final course and exam grades, and reduced dropout rates. Most students thrived under this system, using the flexibility to manage their time effectively without compromising the quality of their work. While a small number of students appeared to struggle with using the policy productively, the overall impact was positive. Overall, flexible token policies show great promise in promoting student success, while also highlighting the importance of providing additional support for students who may need more structure.

Promoting Student Learning by Automatically Managing Resubmission Tokens

Saketh K. Rajesh

(GENERAL AUDIENCE ABSTRACT)

Strict assignment deadlines in university computer science courses can create stress and can unfairly penalize students who face unexpected challenges. One increasingly popular alternative is to give students a small number of "tokens" that allow them to submit assignments late or revise their work. While these policies offer greater flexibility and fairness, they often require instructors to manually track requests and update deadlines across multiple online course platforms, creating a significant administrative burden. This thesis addresses both the practical and educational aspects of flexible deadline policies. First, it introduces a software tool that connects directly to a course's online systems (such as Canvas) and automatically manages the use of tokens, removing the need for instructors to handle requests by hand and allowing students to use tokens without extra steps. Second, the thesis examines the effects of replacing a traditional late-penalty policy with a manually managed, token-based resubmission policy in a large introductory computer science course. The results show that the flexible policy was associated with higher pass rates, better final course and exam grades, and lower dropout rates. Most students used the added flexibility to manage their time more effectively without lowering the quality of their work. Although a small number of students struggled with the increased independence, the overall impact was positive. These findings suggest that flexible token-based policies can support student success while highlighting the importance of additional guidance for students who benefit from more structure.

Dedication

For my parents and brother — my greatest sources of strength and support.

Acknowledgments

I am deeply grateful to my advisor, Dr. Stephen H. Edwards, for his mentorship and steady guidance throughout this work. His feedback and encouragement were invaluable in shaping the direction and success of this research. I would also like to thank my committee, Dr. Bob Edmison, whose support with integration and deployment was critical to the success of this project, and Dr. Clifford A. Shaffer, whose perspective and challenges pushed me to refine my ideas and approach problems in new ways.

This material is based upon work supported by the U.S. National Science Foundation under Grant Nos. 2235337, 2235643, 2235644, 2418493, and 2418494. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

LLM AI tools (Gemini and Claude) were used as a proofreading and editing assistant in this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

CS1 Introductory level undergraduate computer science course

EGP Equitable Grading Practice

LMS Learning Management System

LTI Learning Tools Interoperability

UI User Interface

Chapter 1

Introduction

Traditional late work policies in computer science courses often prioritize rigidity over flexibility. A potential alternative is the token-based system, which provides students with an initial number of tokens that can be used for brief assignment extensions or resubmissions, typically ranging from several hours to a couple of days. This framework introduces flexibility while maintaining clear limits and accountability. These systems have the potential to foster self-regulated learning, reduce student anxiety, and provide an equitable buffer for the unforeseen personal and academic challenges that students inevitably face. By empowering students with agency over their deadlines, instructors aim to create a more supportive and realistic learning environment that mirrors the project management dynamics of the professional world.

However, the pedagogical benefits of token-based policies are often overshadowed by a significant practical barrier: the administrative burden of implementation. In a typical introductory CS course, with hundreds of students, managing a token system manually is a large operational task. Instructors and teaching assistants must meticulously track token usage, field individual extension requests, manually adjust deadlines in the Learning Management System (LMS), and coordinate with external LTI tools for assignments. This process is not only time consuming and prone to human error but also creates a significant barrier to the adoption of these pedagogically sound policies, forcing many educators to revert to simpler, less flexible models out of necessity.

1.1 Problem Statement and Research Contributions

This thesis confronts this critical gap between pedagogical ideals and practical reality. The core problem is twofold: first, the lack of a scalable, automated solution for managing token-based late pass systems creates a significant obstacle for instructors. Second, while the benefits of such systems are often discussed, there is a need for clearer, comparative data that contextualizes their impact against more traditional policies, thereby justifying the effort required to implement them effectively.

To address these interconnected issues, this thesis presents two primary contributions. A Practical Contribution: The design, implementation, and evaluation of a novel software tool, the EGP Broker, that fully automates the management of a token-based late pass system. This tool integrates directly with the Canvas LMS (with considerations to enable adaptability to other LMS, such as Moodle) to handle the administration, tracking, and application of late passes, allowing students to apply extensions to assignments seamlessly. By extending these deadlines to integrated LTI tools, the system provides a robust end to end solution for instructors, making the adoption of a flexible late pass policy feasible even in large-enrollment courses. An Empirical Contribution: A comparative data analysis of student outcomes under two different late work policies in an introductory CS course. By comparing student performance across two semesters, one using a traditional 10% per-day late penalty and another using a manually administered token system that allowed resubmissions, this study observed noticeable improvements in final exam scores and overall course grades under the token-based approach. These results suggest that more flexible late work policies can positively influence student outcomes while introducing new administrative challenges, which motivated the development of the automated system described in this work.

This thesis is organized around two primary areas of focus: an implementation goal and a

research question.

Implementation Goal:

- Develop a system that automates free pass applications across assignment platforms and enables instructors to track their usage.

Research Question:

- In a CS1 course, how does a free pass based system impact student performance and course level outcomes?

1.2 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 reviews relevant literature on late work policies, flexible deadline frameworks, and the balance between accountability and equity in computer science education. It also explores the administrative challenges of implementing such systems, the connections to self-regulated learning, and the motivation for an automated solution. Chapter 3 details the design and implementation of the EGP Broker, describing its user interface, cross-platform considerations, system architecture, and protocol design. Chapter 4 presents the primary features of the EGP Broker from both instructor and student perspectives, illustrating how the system supports flexible late work management in practice. Chapter 5 outlines the comparative study between a traditional late penalty policy and a manually administered token-based policy, including the research context, data collection, analysis procedures, and observed impacts on student performance and course outcomes. Finally, Chapter 6 summarizes the main findings, discusses threats to validity, and identifies opportunities for future research and work.

Chapter 2

Review of Literature

This chapter provides a review of the academic literature surrounding pass-based and token policies in higher education, with a specific focus on the computer science context. It begins by exploring the core philosophical tension that underpins all late policy design: the balance between accountability and equity. Then it presents a taxonomy of common policy models to contextualize the approaches investigated in this thesis. Subsequently, it delves into the pedagogical rationale and empirical evidence for key models, with a particular emphasis on token-based policies. The chapter then examines these policies through the theoretical lenses of Self-Regulated Learning (SRL) and equitable pedagogy, concluding by identifying the specific gaps in current research that this thesis aims to address.

2.1 Accountability vs. Equity

The design of any late work policy is rooted in a fundamental pedagogical debate. The traditional paradigm justifies rigid deadlines and penalties by arguing they prepare students for "real-world" employment and teach responsibility [18]. However, a growing body of contemporary research challenges these assumptions, arguing for policies grounded in equity and student well-being.

This modern perspective reframes "fairness" as "equity," asserting that treating all students the same is only fair if they all begin from the same position and face the same obsta-

cles. Researchers like Hills and Peacock [14] argue that rigid policies uphold the myth of a "normal" student, failing to account for the diverse life circumstances students bring to the classroom [31]. This view highlights the problem of the "shadow policy," where instructors with initially strict rules, privately grant extensions to students assertive enough to ask. This disadvantages those who feel uncomfortable advocating for themselves, a significant barrier identified in multiple studies. Hills and Peacock [14] found that 88% of surveyed students had previously needed an extension but were not comfortable asking for one. Even with a low-barrier automated tool (not requiring a form submission), Benedicto et al. [3] found that significant psychological barriers remain; approximately 46% of students who were uncomfortable requesting an extension cited feelings of "guilt," "shame," or the "fear of being rejected" or perceived as lazy. Consequently, transparent, flexible policies that minimize the need for student self-advocacy are presented as a mechanism to lower these systemic barriers and empower all students.

2.2 Towards Flexible Late Policies

Researchers in computer science education have been investigating structured alternatives to traditional late policies, with token-based systems emerging as one area of exploration. These systems are typically implemented in two forms: late extension tokens and resubmission tokens.

Late extension tokens are the more traditional approach, granting students a set number of "slip days" or "grace tokens" to extend an assignment's due date by a predefined amount of time, such as 24 or 48 hours [5]. This model provides a buffer for students to manage unexpected life events or academic workload conflicts.

Resubmission tokens are used not to extend an initial deadline but to reopen an assignment

for a limited time after a student has received feedback. This approach emphasizes deliberate practice and iterative improvement, providing students with a structured opportunity to correct their work and demonstrate mastery.

There is a growing body of evidence that suggests when given this autonomy, students act responsibly and use the flexibility to enhance their learning. One such systematic implementations of flexible deadline management was documented by Aycock and Uhl [2], who introduced "time banks" allowing students a fixed allocation of penalty-free extension days. Over six years of implementation across three universities with classes ranging from 6-100 students, they observed a notable decrease in student requests for extensions compared to traditional policies. Significantly, their data showed that less than one-third of students used their time bank for each assignment, providing early empirical evidence that students do not abuse flexible systems when given autonomy-a finding that presaged later research on responsible student behavior under flexible policies. Another key study on the "Flextensions" tool built at UC Berkeley,allowing students to request late passes, found that out of 160,013 possible extension opportunities across several large courses, students made only 1,914 requests (less than 2%) [3]. This indicates that students do not abuse the system but rather use extensions when genuinely needed. The reasons cited were primarily legitimate extenuating circumstances, with medical issues (38.0%), family or personal circumstances (19.6%), and managing academic workload (14.0%) being the most common justifications. This pattern of responsible usage aligns with earlier findings by Aycock and Uhl [2], who observed that students "seemed overwhelmingly in favor of the time bank" and used extensions judiciously rather than maximizing their available allocation. Their longitudinal data across multiple institutions provides additional evidence that flexible policies do not encourage irresponsible behavior but rather enable students to manage legitimate conflicts and challenges. This aligns with the findings of Campbell and Reid [5], who also observed

that many students did not maximize their use of extensions, even when they were available without penalty, suggesting that students use them when they feel they are necessary.

Furthermore, students report that the primary benefits of such policies are directly related to learning outcomes and well-being. In a study conducted by Hills and Peacock [14], the most frequently cited benefits were (1) the ability to submit higher-quality work, (2) better management of academic workload, and (3) reduced stress. This is strongly corroborated by findings from the "Flextensions" [19] study, where student feedback was largely positive. Participants reported that the policy had a direct, positive impact on their well-being, learning outcomes, and overall academic experience. Crucially, a significant number of students reported that the policy made them feel valued as individuals by the instructional staff, reinforcing the role in creating a more supportive and equitable learning environment. A key finding from Campbell and Reid [5] supports this: they found a negative correlation between how late an assignment was submitted and the final grade under a punitive policy, but this negative correlation disappeared under flexible policies where extensions were penalty-free. This suggests that when given penalty-free flexibility, students are able to use the extra time to produce higher-quality work.

However, the impact of these policies may not be uniform across all student populations. In a study by Hott [15], the effects of a flexible policy (two no-penalty late days) were compared between a lower-level required course and an upper-level elective. The study found that while students in both courses submitted later under the flexible policy, the outcomes were starkly different. For the upper-level students, performance was similar to or better than under a stricter policy. In contrast, for the lower-level students, overall homework scores were lower with the no-penalty late days. This suggests that students later in their studies may be better equipped to handle the self-regulation demands of a flexible policy, while students in their first or second year may be more vulnerable to its potential downsides, such as

procrastination. This creates a critical tension in the literature. On the one hand, Campbell and Reid [5] findings suggest that flexible policies allow students to produce higher-quality work by removing the negative correlation between lateness and grades. On the other hand, Hott [15] suggest that for students in introductory courses, this same flexibility may lead to lower overall performance. This thesis, with its specific focus on a CS1 course, is uniquely positioned to investigate this tension and provide much-needed data on how a token-based system impacts the performance and behavior of computer science students early in their major or minor completion.

2.3 The Administrative Barrier

Despite the clear pedagogical benefits of token economies, their primary drawback is the significant administrative and infrastructural overhead required to manage them. This challenge has been described as "invisible, uncompensated, and emotionally laborious work" for instructors, who must act as "gatekeepers or judges" for each student's situation. Manually tracking token usage, updating LMS deadlines for individual students, and propagating those changes to external LTI tools is a time-consuming and error-prone process that acts as a major barrier to the adoption of these equitable policies, particularly in large-enrollment courses. Campbell and Reid [5] also note that courses with punitive policies see a higher rate of individual requests for extensions outside of the stated policy, which adds to this administrative burden.

In response, many instructors have developed semi-automated systems to create a structured process for students. A common approach, described by Snider [27] at Ball State University, involves using a "no questions asked" Google Form for students to redeem a pass. Upon a student's submission, the instructor is notified and must then manually deduct the pass

from a non-graded column in the Canvas [16] gradebook and add comments to document the transaction. While this method provides clarity and an official process for students, its limitation is clear: it still places a significant and repetitive manual workload on the instructor for each request to reopen assignments and adjust due dates on a per student basis.

This critical gap between pedagogical desire and practical feasibility has spurred a recent wave of innovation to create fully automated solutions that eliminate this manual burden. At UC Berkeley, researchers developed "Flextensions," a tool to manage a flexible policy in data science courses with as many as 1,500 students. The system leverages a Google Form front-end, but its backend Python script automatically processes requests, validates them against student slip day allowances, and uses the Gradescope API [12] to grant the extension without instructor intervention. This tool was highly effective, reducing the time spent by lead instructors on managing extensions from an estimated 3-5 hours per week to virtually zero [3]. In a similar effort, Oregon State University developed the "Token Tracker," a native Canvas-integrated application[26]. This tool allows instructors to configure a "token bank" for their course, from which students can directly "spend" tokens for predefined adjustments, which the tool then applies automatically within the LMS. The emergence and evolution from manual workflows to these sophisticated, bespoke tools underscore two crucial points: first, the pedagogical value of token policies is widely recognized, and second, there is a critical, unmet need for robust, generalizable, and highly automated infrastructure to make them practical for widespread adoption, especially when courses use external custom assignment platforms.

2.4 Self-Regulated Learning and Equity

Flexible policies, particularly token economies, are often justified through the framework of Self-Regulated Learning (SRL). SRL refers to the process whereby students manage their own thoughts, behaviors, and emotions to successfully navigate their learning journey. By providing a finite number of tokens, instructors require students to strategically plan, monitor their progress, and make conscious decisions about how to allocate a limited resource, all key facets of self-regulation.

The connection between time management and academic success in computer science is well-documented. Research from Shaffer and Edwards [24] at Virginia Tech provides quantitative evidence that students who start programming projects earlier and spread their work out achieve significantly better grades without spending more total time on the task. Their within-subjects comparison revealed that the key differentiator between a student's high-scoring and low-scoring projects was their scheduling, not their total effort. This finding suggests that policies encouraging thoughtful planning over last-minute work can directly foster more effective learning habits. The psychological benefits for students when in a system that increases student control are supported by Aycock and Uhl [2], who noted that their flexible policies helped students "feel at home" with increased workloads in upper-division courses. They connected this to psychological research indicating that "more control in decision-making tends to lower mental strain in workers," suggesting that token-based systems may provide benefits beyond mere deadline flexibility by reducing student stress and increasing sense of agency. Token policies provide a structured mechanism to encourage this kind of strategic planning [14].

2.5 The Need for A Robust Solution

While the literature confirms the value of flexible policies and the challenges of implementing them, two key gaps remain. First, there is a need for more research that directly compares the effects of punitive deduction and token-based policies within the same introductory CS course context to provide a clearer picture of their impact on student behavior and performance.

Second, while the development of custom automation tools at institutions like UC Berkeley [3] and Oregon State University [26] validates the problem, it also highlights the need for more robust and extensible solutions. Many existing tools are tailored to specific infrastructures (e.g., Gradescope) or do not fully address the complex challenge of integrating with the diverse ecosystem of external LTI tools (e.g., auto-graders, interactive textbooks) used in modern CS courses [6] [12].

This thesis is positioned to address these gaps directly. By conducting a multi-semester comparative analysis of these two distinct policies, it provides valuable empirical data on their effects in an authentic CS1 setting. Furthermore, by developing and presenting a novel Canvas plugin tool designed specifically to connect with external assignment platforms and addressing some of the flexibility problems from other automated pass systems (UC Berkeley and Oregon State University) this thesis offers a practical contribution, joining a growing of work aimed at solving the implementation challenges and making flexible, equitable policies more accessible to all educators.

Chapter 3

Design and Implementation of the EGP Broker

The manual administration of token-based late pass policies is time-consuming and not easily scalable. This is a challenge noted by instructors and researchers at UC Berkeley [3] and Oregon State University [26]. In practice, instructors often resort to solutions like manual spreadsheets or simple Google Forms to track token usage. These workflows are cumbersome, error-prone, and create a significant administrative burden that can discourage educators from adopting these pedagogically valuable policies, especially in large-enrollment courses.

To bridge this gap between pedagogical intent and practical reality, this research identified the need for a robust, integrated, and automated tool. Such a tool must solve two interconnected problems simultaneously: 1) eliminate the administrative overhead for instructors, and 2) provide a transparent and seamless interface for students. This chapter details the design and implementation of our solution: the EGP (Equitable Grading Practices) Broker, a centralized tool for managing late passes within a Learning Management System (LMS), and the standardized EGP Protocol designed to connect it with any external assignment platform.

The overarching goal is to create a system that streamlines the entire late pass lifecycle, from configuration and application to tracking and reporting. Achieving this goal requires addressing two distinct but related technical challenges.

3.1 The User Interface Challenge

Currently, there is no accessible, intuitive, and automated user interface within standard LMS platforms, like Canvas, for managing token-based late pass systems. Existing solutions often rely on manual data entry, complex LMS override features, or fragmented external tools that are not designed to communicate with one another. This creates a disjointed experience for both students and instructors.

- For Students: The manual process often requires composing emails, waiting for instructor approval, and manually verifying that deadlines have been changed. This introduces unnecessary stress and uncertainty. A centralized UI, in contrast, empowers students to independently apply a late pass with a single click and receive immediate confirmation, fostering a sense of agency over their academic schedule. By providing a clear dashboard of their allocated, used, and remaining passes, the UI becomes a vital tool for developing time management and self-regulation skills.
- For Instructors: A lack of consolidated data limits instructors' ability to identify assignments where students are consistently struggling. An integrated UI, by centralizing all late pass data, enables instructors to gain actionable insights into class-wide submission patterns, even when assignments are scattered across multiple platforms. This aggregated view facilitates proactive pedagogical adjustments, moving beyond reactive, individual extension requests.

3.2 The Cross-Platform Challenge

Building on the need for a unified interface, modern CS courses increasingly utilize a diverse ecosystem of specialized external assignment platforms (e.g., OpenDSA, CodeWorkout, Web-

CAT) alongside the core LMS. Each of these platforms typically maintains its own internal deadline and submission mechanisms. A lack of a standardized and machine-readable protocol prevents the automated propagation of extensions from a central broker to these external systems [25][10][4].

This forces instructors into time-consuming, manual adjustments across multiple platforms for a single student on a single assignment. To solve this, this research proposes a simple, adaptable, and platform-agnostic EGP Protocol. This protocol ensures that any external assignment platform that implements its requirements can receive extension information automatically whenever a student applies a late pass to an LTI-linked assignment in the LMS. This ability to seamlessly pass extension information to a wide array of external platforms is the major distinguishing factor between the EGP Broker/Protocol and other attempts at automating late pass management, such as UC Berkeley's Flextensions [3], and Oregon State's Token Tracker [26], which are often designed for integration with a specific tool like Gradescope [12] or Canvas Assignments.

The remainder of this chapter details the technical implementation of this system. Section 3.2 outlines the overall System Architecture, describing the key components and their interactions. Section 3.3 specifies the Technology Stack used for development. Section 3.4 walks through the implementation of the EGP Broker, including the student and instructor user interfaces. Finally, Section 3.5 provides a detailed specification of the EGP Protocol.

3.3 System Design and Architecture

3.3.1 High-Level Overview

The EGP Broker is architected as a modern, decoupled web application designed to function as a trusted intermediary between a Learning Management System (LMS), students, instructors, and various third-party educational platforms. At its core, the system is a Learning Tools Interoperability (LTI) 1.3 compliant tool that operates as a Single-Page Application (SPA). This architecture was chosen to provide a responsive, seamless user experience while ensuring secure, standardized communication with the host LMS (in this case, Canvas). The entire system is built upon an Identity and Access Management (IAM) model, ensuring that all functionality and data are served according to each user's specific role and privileges (e.g., student, instructor).

The frontend is built using the React library [20], which allows for a dynamic and interactive user interface. This frontend communicates with a dedicated backend service built on Express.js. The backend is responsible for several critical functions: orchestrating the secure LTI 1.3 launch and authentication process, managing all business logic related to late pass calculation and application, persisting data, and interacting with external assignment platforms via a standardized endpoint defined by the EGP Protocol. To manage the complexities of the LTI 1.3 standard, the backend leverages the ltijs library, which handles the OAuth 2.0 and OpenID Connect flows required for secure communication, including the fetching and management of LMS-specific access tokens [1] [13].

3.3.2 Core Components and Data Flow

The system is composed of three primary interacting components: the LMS, the EGP Broker, and external assignment platforms. The diagram below illustrates the relationships and data flow between them. This can be seen in Figure 3.1.

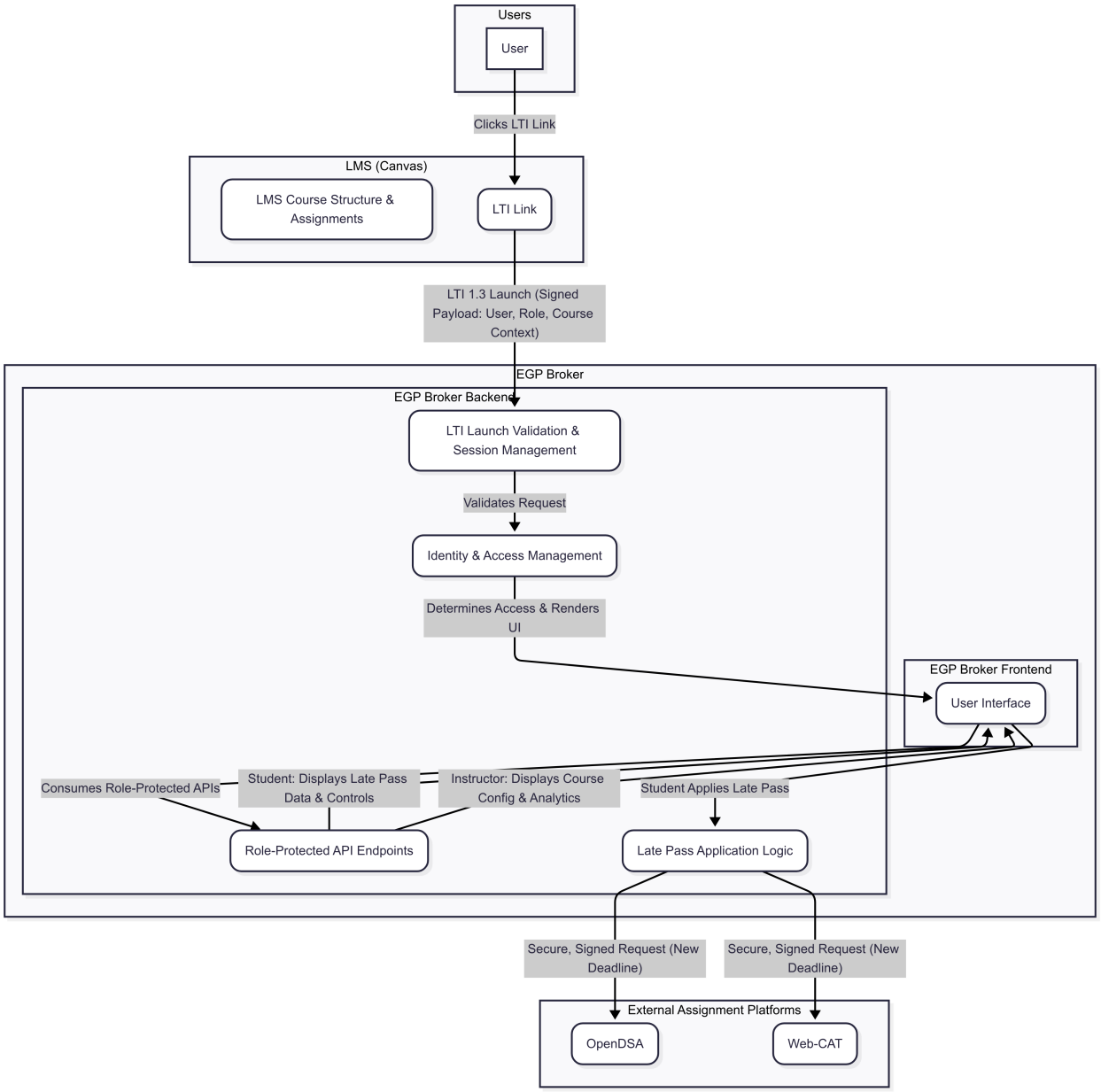


Figure 3.1: EGP Broker Core Components Interaction

1. Learning Management System (Canvas): The LMS serves as the entry point for all users. It houses the course structure, assignments, and the LTI link that launches the EGP Broker. When a user clicks this link, Canvas initiates the LTI 1.3 launch flow, sending a signed payload containing user identity, role, and course context to the EGP Broker.

2. EGP Broker: This is the central hub of the system. Based on the role provided during the LTI launch, the EGP Broker's identity and access management layer determines what resources the user can access.The backend receives the LTI launch, validates the request, and establishes a secure session. It exposes a set of role-protected API endpoints for the frontend to consume. Its frontend renders the appropriate user interface based on the user's role. These endpoints are secured by user LTI keys. For users with student privileges, it displays personal late pass data and application controls. For users with instructor privileges, it provides access to course-level configuration and analytics dashboards.

3. External Assignment Platforms (e.g., OpenDSA, Web-CAT): These are the third-party tools where students complete their work. When a student applies a late pass via the EGP Broker, the EGP Broker's backend sends a secure, signed request to the relevant external platform's endpoint, informing it of the new, extended deadline for that specific student and assignment.

3.3.3 The Role of LTI 1.3

LTI 1.3 is the foundational technology that enables the EGP Broker to integrate seamlessly and securely with the LMS. It provides a standardized specification for establishing trust and exchanging information between a "platform" (the LMS) and a "tool" (the EGP Broker).

When a user launches the EGP Broker from Canvas, the LTI 1.3 protocol orchestrates an OAuth 2.0 and OpenID Connect (OIDC) handshake [1]. This process achieves two critical objectives without requiring users to create a separate account or log in to the EGP Broker:

1. Authentication: The OIDC layer verifies the user's identity and provides the EGP Broker with essential information, such as their name, unique ID, and (most importantly) their role within the course (e.g., Instructor, Student). This role information forms the basis of the EGP Broker's internal identity and access management system.

2. Authorization: The OAuth 2.0 layer provides the EGP Broker with a secure, short-lived access token. This token grants the EGP Broker the necessary permissions to act on the user's behalf, such as accessing assignments and free passes within the EGP Broker.

By leveraging this robust framework, the EGP Broker becomes a portable and interoperable tool that can be integrated into any LMS that supports the LTI 1.3 standard (e.g., Moodle, Blackboard), ensuring a consistent and secure user experience across different educational ecosystems.

3.3.4 User Authentication with LTI 1.3

A deliberate design choice in the EGP Broker was to delegate authentication, role assignment, and user metadata to the LMS (Canvas) through the LTI 1.3 standard, rather than implementing a separate username password authentication system. This decision carries important implications for usability, interoperability, and security.

Advantages of Leveraging Canvas

Relying on Canvas (the LMS) for authentication and role management provides several important advantages. First, it offers a seamless user experience. Students and instructors can access the EGP Broker directly through the LMS without the need to create or manage separate accounts. This reduces cognitive load, minimizes login friction, and encourages adoption. In addition, Canvas provides authoritative role and context information, including course membership, user roles (Instructor, Student, TA) and contextual identifiers. This metadata is essential for enforcing fine-grained access control within the EGP Broker. Security is also strengthened by delegation. By deferring identity management to Canvas, the EGP Broker inherits the institution's enterprise-level protections, such as multi-factor authentication and single sign-on, which would be difficult to replicate in a custom system. Because these credentials and role attributes are delivered through standardized LTI payloads, the EGP Broker remains portable across any LMS that supports LTI 1.3, ensuring interoperability of core components beyond Canvas with minimal changes.

Tradeoffs and Limitations

This design decision also introduces notable limitations and tradeoffs that were carefully considered. The most immediate is the dependency on LMS availability. The EGP Broker cannot function in isolation as it relies on the LMS to provision user identities. If Canvas is unavailable or misconfigured, users are unable to access the system. However, in practice, this risk is somewhat mitigated, depending on the structure of the course. For example, in CS 1114 at Virginia Tech, many assignments are already launched through Canvas integrations with external tools, so students depend on Canvas regardless of the EGP Broker's design. As such, the EGP Broker does not introduce a new point of failure so much as it aligns with

existing workflows.

Another drawback is limited metadata control—the EGP Broker is constrained to the structure and role definitions that Canvas provides, which may restrict flexibility in cases where custom roles or additional attributes are needed. Implementing the LTI 1.3 stack also brings added complexity compared to a simple password login, requiring careful handling of cryptographic validation, JWTs, and OAuth 2.0 exchanges. This complexity was partially mitigated by the use of the lti.js library, which provides a higher-level abstraction over the LTI 1.3 launch sequence. The library automates many of the repetitive validation steps, such as verifying JWT signatures and exchanging OAuth tokens, allowing the development effort to focus on application logic rather than low level protocol details [8].

Finally, this choice reduces autonomy, as the EGP Broker cannot independently onboard users outside the LMS ecosystem-such as guest researchers, or external evaluators- without additional workarounds. In practice, the only role that did not align directly with Canvas's course-based model was the administrator. Unlike instructors or students, who are always tied to a specific course, administrators in Virginia Tech's case have system-wide privileges that span across courses and extend beyond the scope of a single class. This discrepancy required special handling, since the LTI launch payload is designed for course-scoped roles. During the design process, this tradeoff was accepted because the primary use case of the EGP Broker is centered on students and instructors within a course. Admin functionality was treated as an special case that could be layered on later as an exception to LTI authentication, rather than a core requirement of the initial design.

Overarching Factors

In general, the reliance on Canvas for authentication and roles reflects a design philosophy of interoperability and security through standardization, at the cost of reduced independence from the LMS. At the same time, this approach significantly simplified the implementation of identity and access management within the EGP Broker. Because Canvas already enforces a well-defined role hierarchy and strong authentication mechanisms, the EGP Broker was able to inherit these guarantees rather than reinvent them. This meant that critical tasks-such as distinguishing between instructors and students, enforcing access policies, and ensuring that only authenticated users could reach sensitive endpoints-were largely handled through the LTI launch payload. As a result, the EGP Broker's IAM layer could remain lightweight and focused on interpreting Canvas-provided claims, rather than managing an entirely separate account system. The ease of leveraging these built-in roles and security assurances ultimately accelerated development, reduced the likelihood of introducing vulnerabilities, and ensured that the system remained aligned with institutional standards from the outset.

3.3.5 Security Model

Given that the system handles sensitive student data and has privileged access to both the LMS and external assignment platforms, a multi-layered security model is essential. Authentication and authorization are managed at two primary interfaces: the connection between the LMS and the EGP Broker, and the connection between the EGP Broker and external platforms.

LMS-to-Broker Security

This interface is secured by the LTI 1.3 standard. All communication is initiated through a trusted, pre-configured handshake. The LTI launch provides the EGP Broker with a trusted, verifiable assertion of the user's role, which underpins the entire identity and access management policy within the application. Data integrity and authenticity are ensured through the use of JSON Web Tokens (JWTs) signed with asymmetric key pairs (RS256), preventing tampering and ensuring all launch requests originate from the legitimate LMS platform.

Broker-to-External Assignment Platform Security

To ensure secure communication between the EGP Broker and external assignment platforms, the EGP Protocol mandates the use of the OAuth 2.0 Client Credentials grant type. When an administrator first links an external platform to their course via the EGP Broker, the platform is issued a unique client_id and client_secret. To make a request (e.g., to query for an extension), the EGP Broker must first exchange these credentials for a short-lived access token from the external platform's authorization server. This token is then included in all subsequent API calls as a bearer token. This ensures that the external platform only accepts requests from the legitimate EGP Broker, preventing unauthorized access or manipulation of student deadlines.

Role-Based Access Control

The roles defined and provisioned by Canvas through the LTI 1.3 protocol form the foundation of role-based access control (RBAC) throughout the EGP Broker. When a user launches the EGP Broker from Canvas, their authenticated role is embedded within the LTI launch

payload and verified through the JWT signature validation process described above. These verified roles are then enforced at the application level to gate access to both web interface components and API endpoints. Students who authenticate through the LMS are restricted to student-only web pages and can only invoke student endpoints on the EGP Broker to execute students specific functionality, such as using a pass on an assignment or viewing their own past pass usage. Similarly, instructors are limited to instructor-specific functionality, including course management, student data oversight, and extension approval capabilities. By grounding RBAC in the LTI 1.3 assertion, the EGP Broker ensures that access control decisions are tied to the authoritative, institutional role hierarchy maintained by Canvas, rather than relying on locally-managed role assignments that could become inconsistent with the LMS. This approach provides both security and consistency. Users cannot escalate their privileges within the EGP Broker without corresponding changes in Canvas, and role synchronization happens automatically with each LTI launch, ensuring that access rights remain current as course rosters change.

3.4 Technology Stack

The selection of technologies for the EGP Broker was guided by the principles of robustness, scalability, and ease of development within a modern web ecosystem. The stack leverages a full-stack JavaScript approach, which streamlines development by allowing for a consistent language across both the client and server. The EGP Broker also uses the No-SQL Database, MongoDB, to store student, instructor, course, and pass information.

3.4.1 Frontend

The user interface is a Single-Page Application (SPA) built with React. This choice was motivated by React's component-based architecture, which facilitates the creation of a modular, maintainable, and interactive UI. By managing its own state, the frontend can dynamically render views for students and instructors without requiring full page reloads, leading to a faster and more fluid user experience [20].

3.4.2 Backend

The backend server is built on Node.js [23] using the Express.js framework [28]. Node.js was selected for its non-blocking, event-driven architecture, which is well-suited for handling concurrent requests from the LMS and external tools. Express.js provides a minimalist and flexible foundation for building the RESTful API [11] that serves the frontend and communicates with other services.

A critical component of the backend is the LTIjs library. This library abstracts away the significant complexity of the LTI 1.3 standard's security model [7], handling the OAuth 2.0 and OIDC handshakes, token management, and cryptographic signing of messages. The use of LTIjs significantly accelerated development by allowing the focus to remain on the core application logic rather than the low-level details of the LTI protocol [8].

3.4.3 Database

The diagram, shown in Figure 3.2, illustrates the logical structure of the system's MongoDB [21] schema. Unlike a traditional relational database, this schema is modeled around document-based data organization, where entities are represented as flexible, self-contained

documents rather than rigidly normalized tables. Relationships such as those between courses, instructors, and students are expressed through references, while repeated or context-specific data (such as pass usage within enrollments) is embedded directly within documents to reduce query complexity and improve performance. This hybrid approach maintains conceptual clarity through entity relationships while leveraging MongoDB's strengths in scalability, flexibility, and hierarchical data representation, making it well suited for evolving educational workflows and course-specific state management.

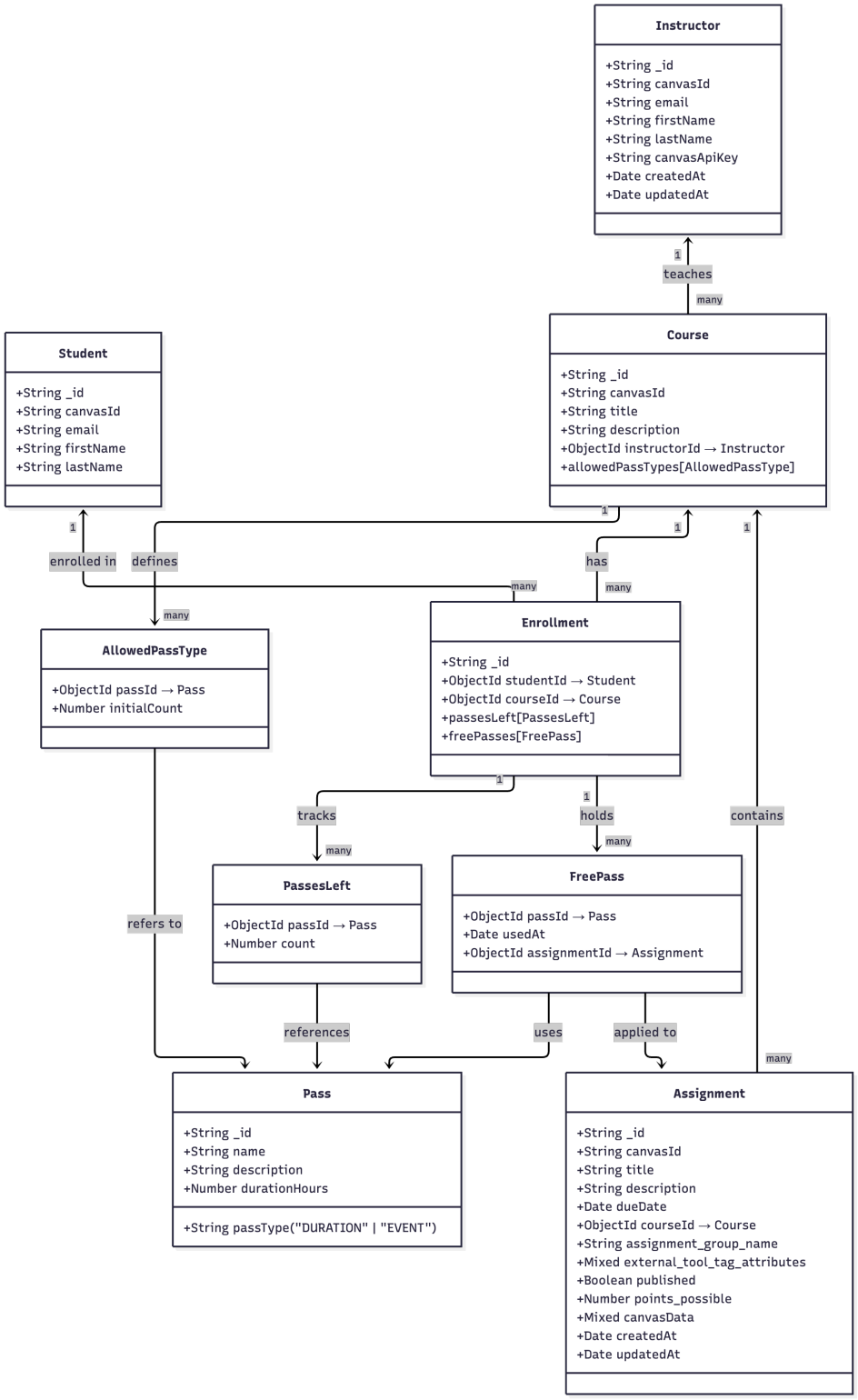


Figure 3.2: Database Schema

- **Pass:** The Pass entity defines reusable, standardized units of flexibility that control student privileges within courses. Each pass has a passType field, which can be either DURATION for time-based extensions or EVENT for single-use exceptions. When applicable, fields such as durationHours are included to describe how long a pass extends a deadline. The purpose of this entity is to provide a consistent definition of what kinds of passes exist within the platform, which courses can adopt, and how they behave when used.

- **PassType (Embedded in Course):** Within each course document, the allowedPassTypes field defines which pass types are valid for that course. Each entry references a corresponding Pass and specifies the initial number of passes each student receives upon enrollment. This embedded structure allows instructors to tailor pass policies per course while preserving a link to the global pass definition.

- **Instructor:** The Instructor document represents course owners and managers. It includes identifiers, email, name, and optional LMS API credentials, along with creation and update timestamps. Instructors are referenced in courses via the instructorId field. This structure supports role-based access, ensuring instructors can view and manage all students and their pass usage within their courses.

- **Student:** The Student document stores identifiers, names, and contact information for learners. It is intentionally separated from course and enrollment data to ensure strong privacy boundaries, students can only access their own data. This separation also simplifies authentication and role-based query logic, supporting scalable, multi-course participation.

- **Course:** The Course document stores metadata such as course title, description, and Canvas identifiers. It references an instructor and embeds allowed pass configurations

through allowedPassTypes. This structure allows each course to define its own policies for passes while maintaining consistency across the platform through shared pass definitions.

- **Enrollment:** The Enrollment document connects a student to a specific course and captures the dynamic state of their participation. It includes:
  - passesLeft: a subdocument that tracks how many passes of each type the student still has available.
  - freePasses: a list of subdocuments, each recording a pass the student has used, storing details such as the passType, the assignmentId it was applied to, and the timestamp of use.

  When a student uses a pass, the system automatically deducts one from the corresponding passesLeft count and appends a new entry to freePasses. This entry provides a complete record of which passes were used, when, and for what purpose, enabling detailed auditing and tracking of student flexibility.

- **Assignment:** The Assignment document represents individual course tasks or assessments. Each assignment belongs to a course and includes metadata such as title, due date, points possible, and optional Canvas fields. Assignments can reference pass usage when extensions or exemptions are granted, ensuring that pass-related actions are reflected consistently in grading and scheduling workflows.

Instructors create courses and define their allowed pass types. Students enroll in these courses, receiving initial allocations of passes stored in their passesLeft subdocument. As students use passes, the system updates both their passesLeft and freePasses, ensuring accurate tracking of remaining flexibility and usage history. Assignments may reference used

passes for deadline adjustments, while instructors maintain oversight of all pass activity within their courses.

3.4.4 Development and Deployment

To ensure a consistent and reproducible development environment, the entire application is containerized using Docker [9]. This allows developers to run the full stack (frontend, backend, and database) locally with a single command, eliminating "works on my machine" issues. This allows for easy portability and ability for multiple people to get started with development quickly now and in the future.

For testing and integration with a live LMS instance, ngrok was used. Ngrok creates a secure public URL [22] that tunnels requests to the local Docker container. This was essential for development, as it allowed the tool running on a local machine to be installed and tested within a development Canvas instance hosted on a Virginia Tech Kubernetes [30] cluster, enabling rapid iteration and validation of the LTI integration in a realistic environment.

3.5 EGP Protocol Design

The operational efficacy of the EGP Broker, particularly its ability to ensure consistent policy application across heterogeneous assignment platforms, is based on the EGP Protocol. This protocol defines the "rules of engagement" for a three-part conversation between the EGP Broker, the external Assignment Platform, and the primary Learning Management System (LMS). The EGP Protocol is a platform-agnostic standard designed to enable personalized, per-student deadline adjustments that can be uniformly recognized across any compliant external system. At a high level, the EGP Broker acts as the central coordinator: it validates

a student's request, communicates the extension to the external Assignment Platform, and, upon success, informs the LMS to create an official deadline override. This process formalizes a critical interoperability layer that addresses the prevalent fragmentation of deadline management in multi-platform courses, establishing a clear contractual agreement for seamless integration.

3.5.1 Adaptability and Scope

A key design objective for the EGP Protocol is to establish broad compatibility with the diverse landscape of external assignment platforms. The protocol is engineered to accommodate two primary categories of assignment platforms,internal deadline tracking and LMS dependent score reporting only platforms.

Self-Contained Deadline Systems

Many platforms (such as OpenDSA and Web-CAT) were developed with their own robust internal due date managers. These systems operate with their own internal deadlines. The EGP Protocol provides a standardized, machine-readable bridge for the EGP Broker to communicate with and control these isolated deadline mechanisms, propagating extensions directly to their authoritative source without requiring a complete architectural overhaul of the platform.

LMS-Dependent System

Simpler or more targeted tools may function primarily as content delivery and score-reporting mechanisms, relying on the LMS (e.g., Canvas) as the authoritative source for deadlines. The EGP Protocol is also compatible with this architecture. In this scenario, the external

tool must still implement the protocol's API to confirm assignment existence, but the EGP Broker's primary role shifts to coordinating with the Canvas API's override feature to grant the extension, while still tracking pass usage centrally. This dual compatibility ensures the EGP Broker can serve as a universal late pass management solution across varied technical contexts.

End-to-End Protocol Workflow

The following sequence diagram and description illustrate the complete, end-to-end workflow when a student applies a late pass using the EGP Broker. This process ensures that deadlines are updated consistently across the EGP Broker's internal state, any external assignment platforms, and the master LMS gradebook (see Figure 3.3).

1. Initial Validation: The frontend sends a request to the EGP Broker's backend. The backend first validates the request by querying the Enrollments collection to ensure the student has a sufficient pass balance. If the request is invalid, it returns an error to the user.

2. External Platform Communication: If the initial validation succeeds, the EGP Broker determines if the assignment is hosted on an EGP Protocol compliant external platform with a self-contained deadline system.

    - If so, it sends a POST /egp_broker/student_extensions request to that platform's API endpoint. The external platform then updates its internal deadline for the student and returns a success message. If this request fails, the entire transaction is aborted, and an error is shown to the student.
    - If the assignment is hosted on an LMS-dependent platform or a LMS native assignment or quiz, this step is skipped.

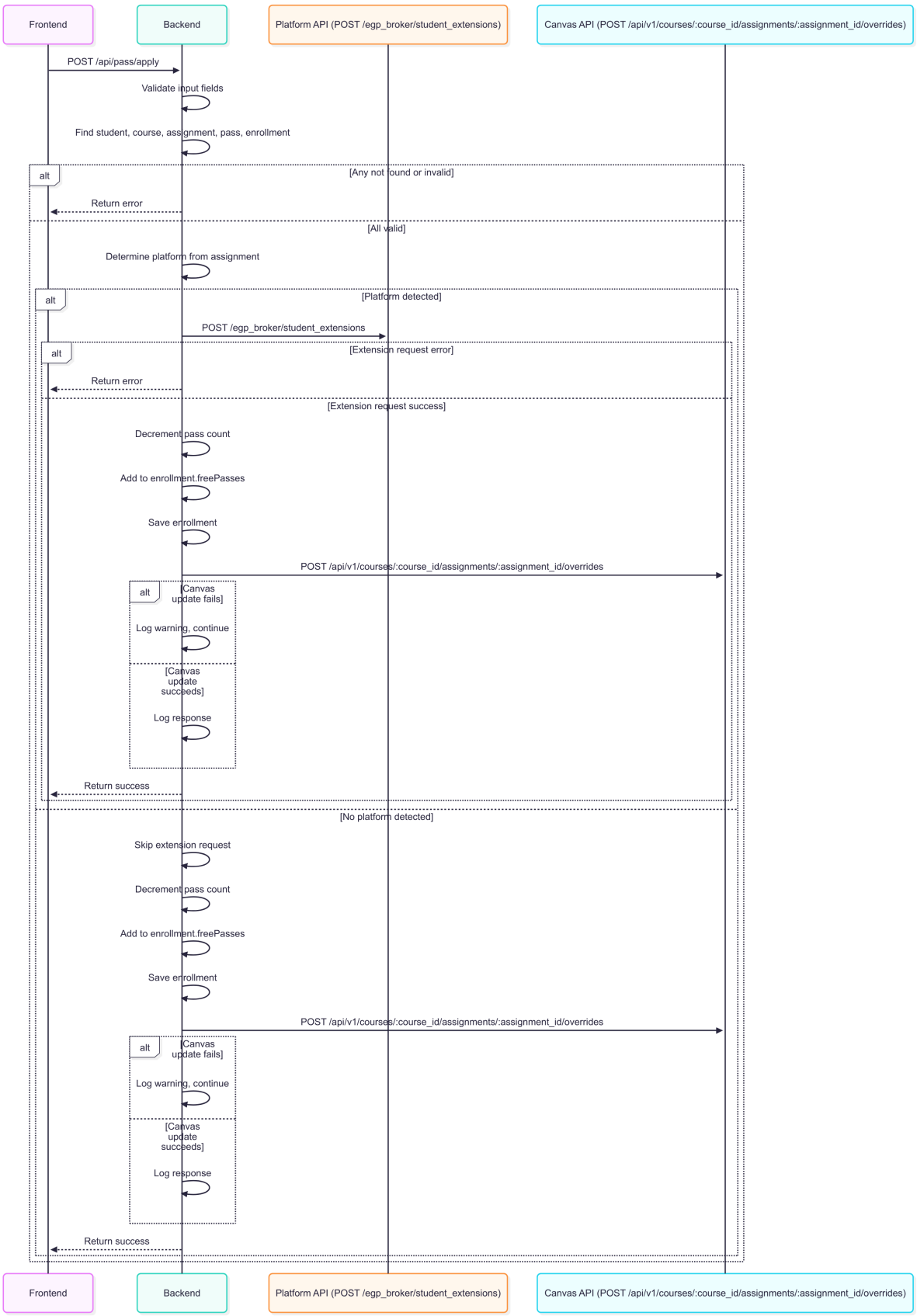


Figure 3.3: EGP Protocol Full Flow

3. Internal State Update: Upon successful confirmation from the external platform (or if the previous step was skipped), the EGP Broker's backend updates its own internal state. It decrements the student's pass count in their Enrollments document and creates a new entry in the Logs collection to provide a clear audit trail.

4. LMS Override: To ensure the official course gradebook reflects the extension, the EGP Broker performs the final step: it makes an API call to the LMS, in the case of Canvas (POST /api/v1/courses/:course_id/assignments/:assignment_id/overrides). This request creates a student-specific assignment override, which officially changes the due date for that student within the Canvas interface and gradebook.

Internal Platform Requirements

The EGP Protocol mandates that a compliant external platform expose a single, standardized RESTful API endpoint to receive extension information from the EGP Broker. Communication is secured using the OAuth 2.0 Client Credentials flow, where the EGP Broker uses a pre-shared client_id and client_secret to obtain a bearer token for authenticating its requests.

POST /egp_broker/student_extensions

This endpoint serves as the primary mechanism for the EGP Broker to issue or modify an extension. It supports an "upsert" (create or update) operation, ensuring idempotent management of student extensions. The request body is a JSON object containing three key pieces of information: the student's identity, the assignment's identity, and the details of the pass being applied.

Example Request Body

```
{
    "student_info": {
        "first_name": "Jane",
        "last_name": "Doe",
        "email": "jane.doe@example.com",
        "lms_id": "123456"
    },
    "assignment_info": {
        "external_tool_url": "https://opendsa.cs.vt.edu/
            lti/book/chapter/assignment.html",
        "lms_id": "78901"
    },
    "pass_info": {
        "duration_hours": 24
    }
}
```

For an external platform with its own internal deadline manager to be fully compliant with the EGP Protocol, it must satisfy specific internal architectural requirements. These ensure that the platform can correctly interpret and enforce the extensions granted by the EGP Broker.

1. Student Identification: The platform must be able to uniquely identify a student based on the information provided in the student_info object. While multiple fields are provided for flexibility, the lms_id is the most robust and ideal identifier. It is guaranteed to be unique within the LMS instance and is immutable, unlike a name or email address, which a student might change.

2. Assignment Identification: The platform must be able to identify the specific assign-

ment to be extended using the assignment_info object. The most reliable method is to use the lms_id of the assignment. Alternatively, the platform can parse the external_tool_url, which is the deep-linked URL configured in the Canvas assignment. This URL will typically contain a unique internal identifier (e.g., a path or query parameter) that the platform can extract to locate the correct assignment.

3. Student-Specific Deadline Storage: The platform must have a mechanism to store and apply deadlines on a per-student, per-assignment basis, overriding any global or course-wide default due dates. The implementation of this can vary. A common approach is to create a dedicated "student extensions" table in the platform's database. For example, in the integration with OpenDSA, a new table was created to store records consisting of a student_id, an assignment_id, and a new_due_date. When a student accesses an assignment, the system first checks this table for an override before falling back to the default deadline.

Chapter 4

EGP Broker Features and User Interface

The preceding sections detail the conceptual architecture, protocol, and technology stack. This section describes the concrete implementation of the EGP Broker itself, focusing on the user-facing interfaces for students and instructors. The design of these interfaces is driven by the goal of providing a simple, transparent, and powerful user experience.

4.1 Instructor User Interface

The instructor interface provides a comprehensive suite of tools for configuring the late pass policy and monitoring its usage. The workflow begins with a one-time setup process and then transitions to a multi-faceted dashboard for ongoing course management. The EGP Broker tool is installed as an external tool in the "course_navigation" section of Canvas (or can be installed in an equivalent section in other LTI 1.3 compliant LMSs).

4.1.1 Initial Course Setup

When an instructor launches the EGP Broker in a course for the first time, they are guided through a simple, two-step setup process, as shown in Figure 4.1.

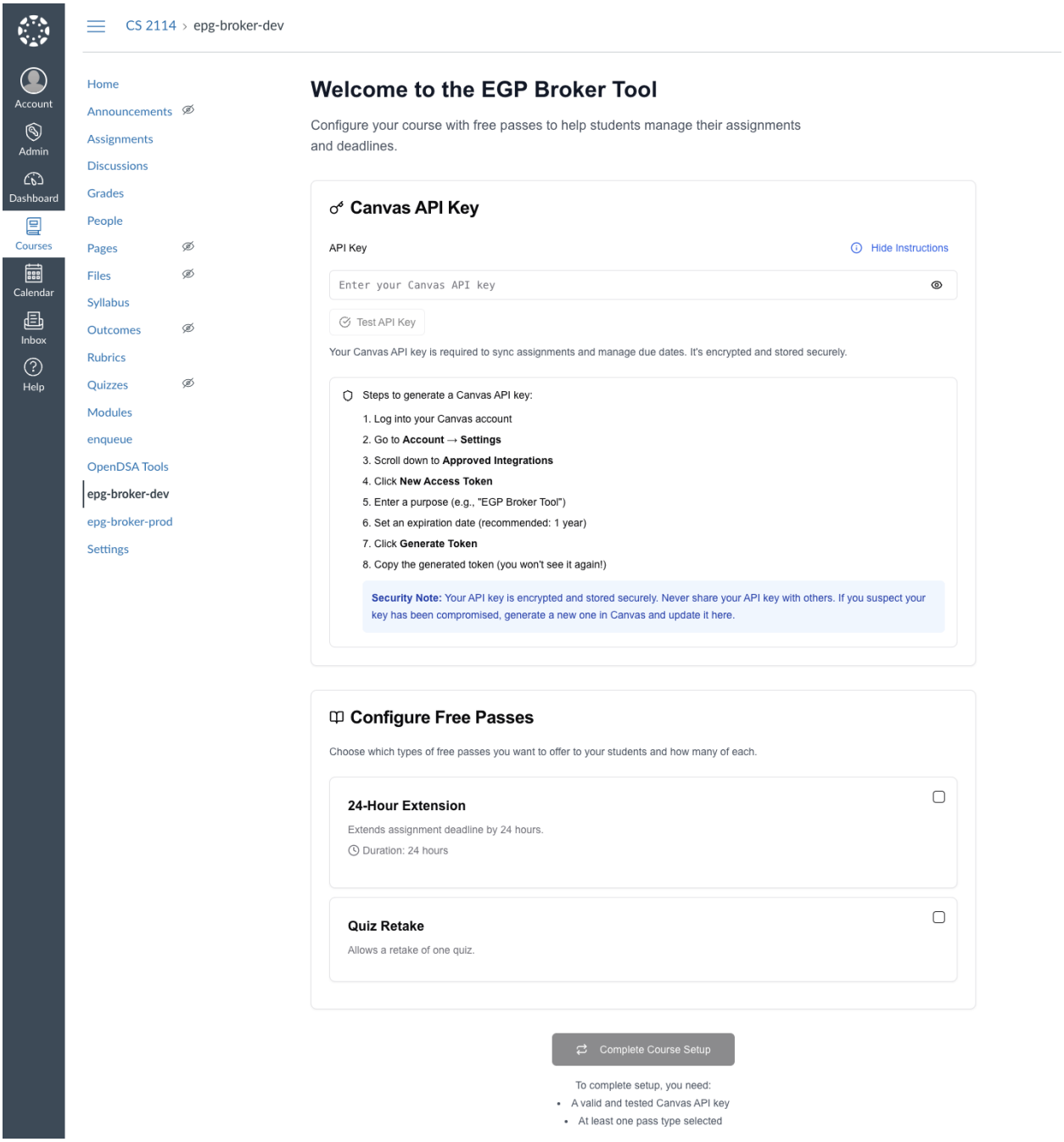


Figure 4.1: Initial Course Setup

1. Canvas API Key Configuration: The first step requires the instructor to provide a Canvas API key. This key is essential for the EGP Broker to perform actions on the instructor's behalf, such as syncing the assignment list and creating deadline overrides. The interface includes clear instructions on how to generate a key in Canvas and a "Test API Key" button that provides immediate feedback, confirming whether the key is valid and has the necessary permissions.

2. Pass Policy Configuration: Once a valid API key is provided, the instructor configures the course's late pass policy. They can choose from the globally defined pass types (e.g., "24-Hour Extension," "Quiz Retake") and use simple controls to set the initial number of each pass type that every student will receive upon enrollment. After completing this step, the course is fully registered in the EGP Broker system.

4.1.2 Instructor Dashboard

After setup, the main entry point for an instructor is the Dashboard tab (see Figure 4.2). This view provides a high-level, at-a-glance summary of the course and late pass usage.

1. Key Metrics: At the top of the page, a series of cards display key statistics, including the total number of enrolled students, the total number of passes used across the entire course, the number of passes used in the last seven days, and the average number of passes used per student. These metrics allow instructors to quickly gauge overall trends.

2. Course Assignments: The main body of the dashboard lists all course assignments, grouped by their Canvas assignment group. This provides a familiar structure and allows instructors to see which assignments are eligible for passes.

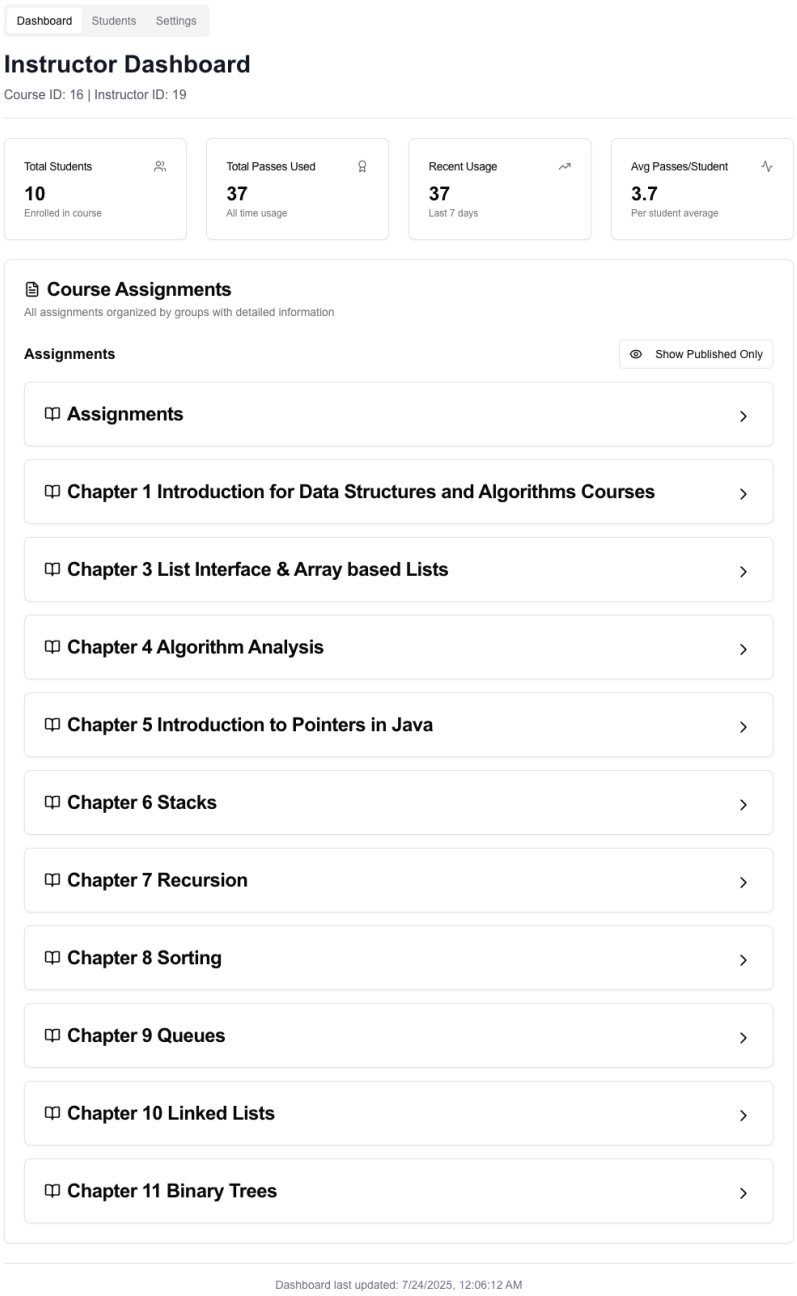


Figure 4.2: Instructor Dashboard

4.1.3 Student Management View

The Students tab (Figure 4.3) provides an interface for monitoring and managing individual student pass usage.

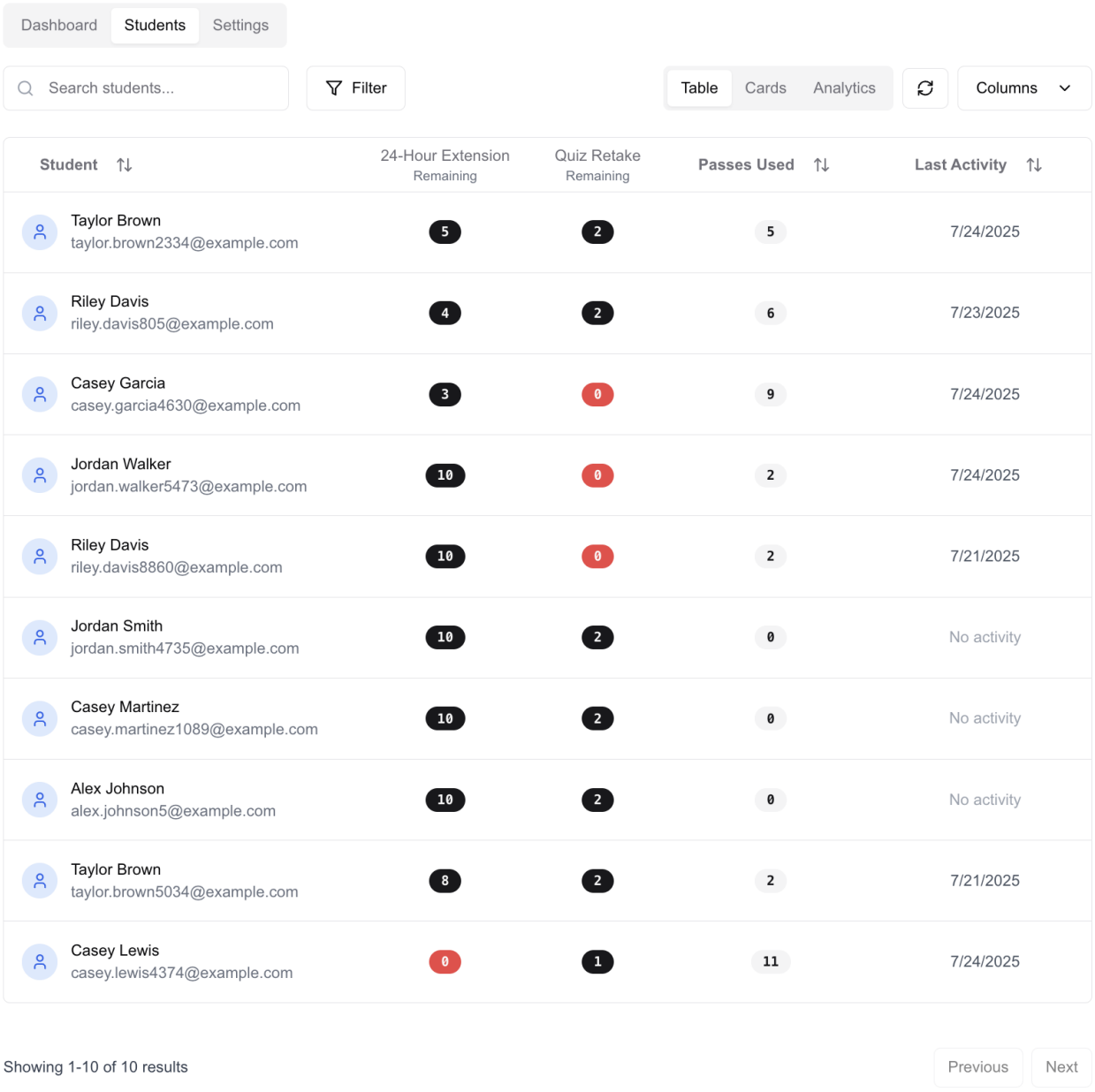


Figure 4.3: Student Management Tab

Student Roster (Figure 4.3): This view displays a searchable and filterable table of every

student enrolled in the course. Each row shows the student's name and their remaining balance for each type of pass available in the course, providing a quick way to see who has used their passes and who has not.

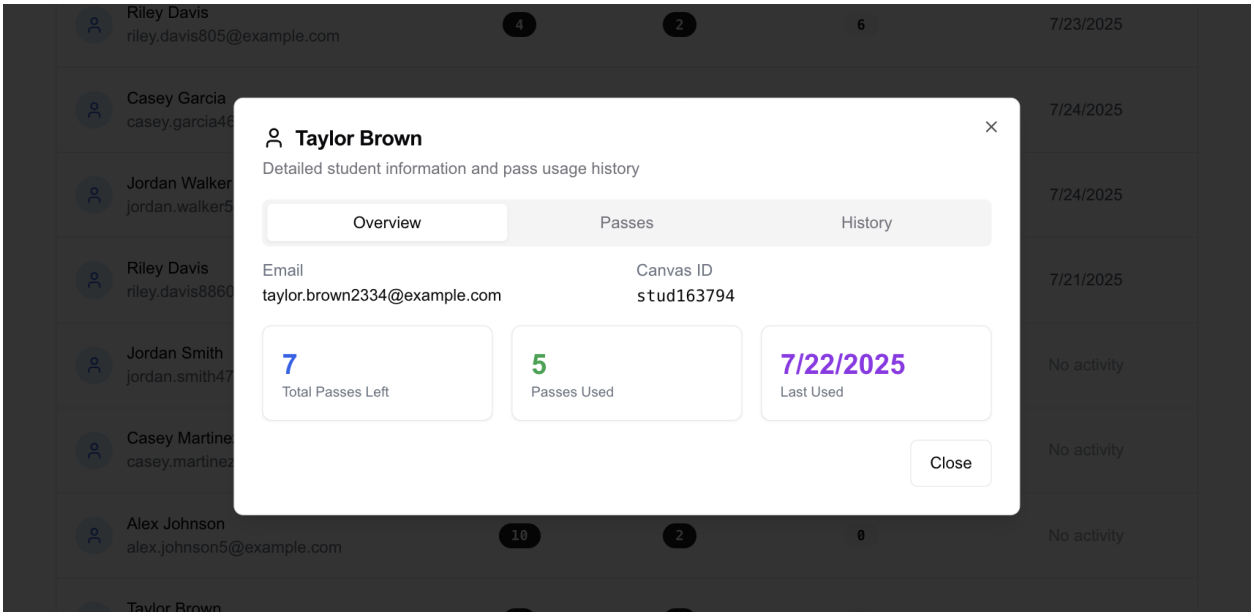


Figure 4.4: Student Information Modal: Overview

Detailed Student View (Figure 4.4, Figure 4.4): Clicking on any student in the roster opens a modal window with a more detailed breakdown of their activity. This detailed view has three tabs:

- Overview: Shows a summary of the student's total passes left, total passes used, and the date of their last activity.

- History: This tab provides a complete, reverse-chronological log of every single pass the student has used, including the name of the assignment it was applied to and the exact timestamp of the transaction. This provides a clear and unambiguous audit trail.

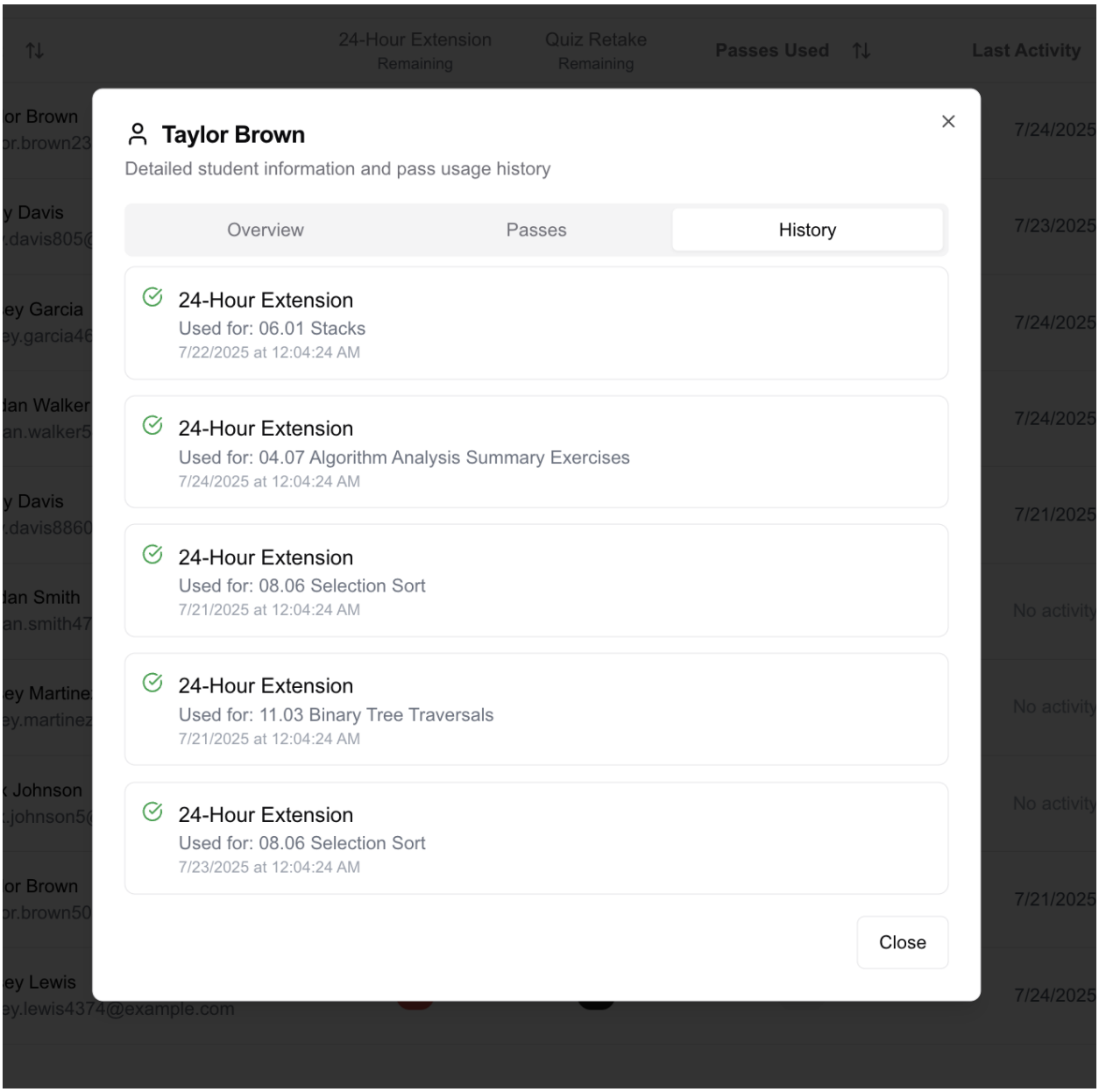


Figure 4.5: Student Information Modal: History

4.1.4 Instructor Settings

The Settings tab (Figure 4.6) allows an instructor to manage their course configuration after the initial setup. The primary function of this page is to allow the instructor to update or replace their Canvas API key if it expires or is compromised, ensuring the EGP Broker can continue to function correctly.

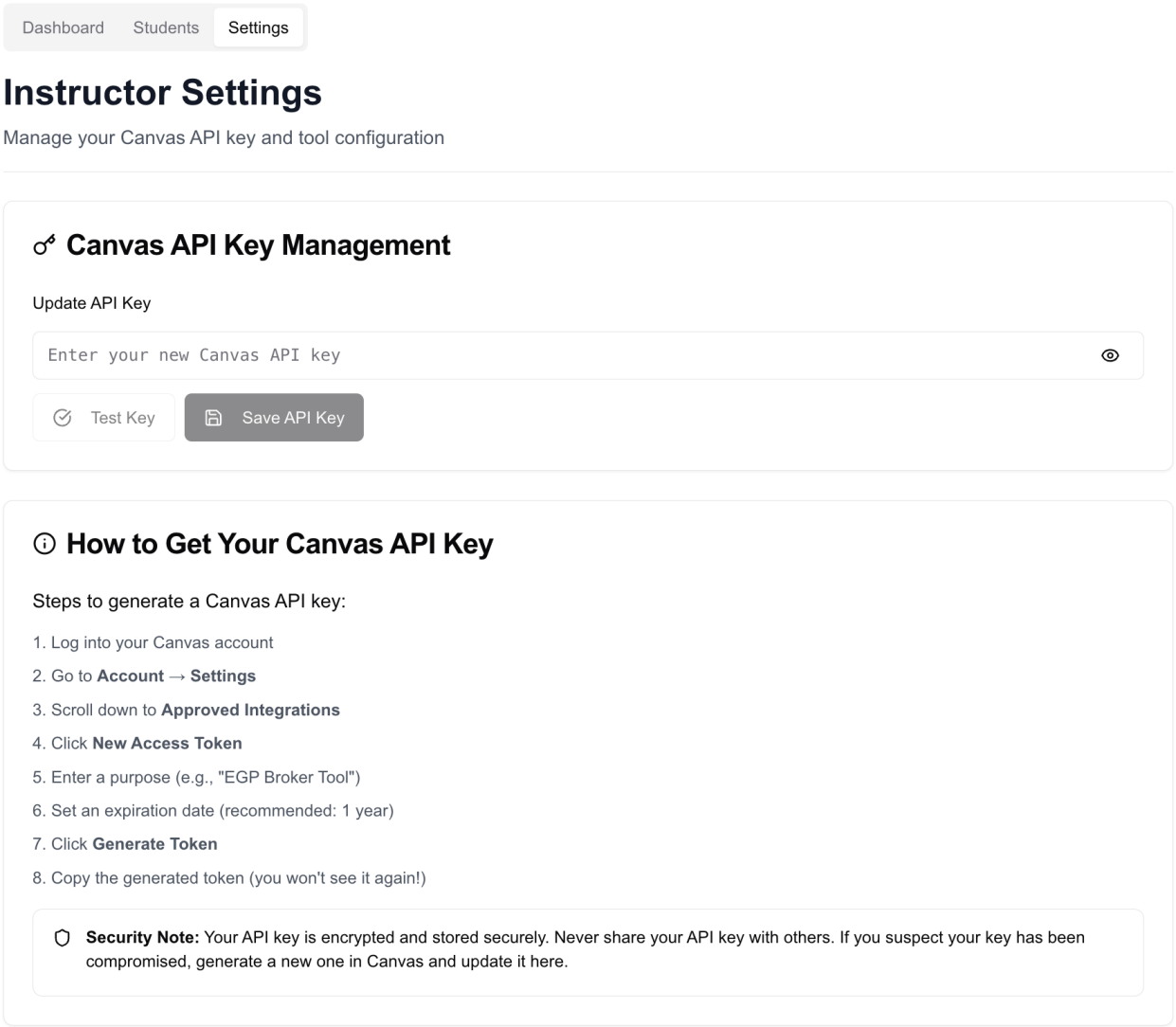


Figure 4.6: Instructor Settings

The Settings tab allows an instructor to manage their course configuration after the initial

setup. The primary function of this page is to allow the instructor to update or replace their Canvas API key if it expires or is compromised, ensuring the EGP Broker can continue to function correctly.

4.2 Student User Interface

The student interface Figure 4.7 is designed for clarity and ease of use, empowering students to manage their own deadlines with minimal friction. Upon launching the tool from an LTI link in Canvas (button is located in the "course_navigation" section) [17], the student is presented with a clean and simple, dashboard-style view that immediately provides all necessary information.

1. Available Passes: At the top of the interface, a prominent "Available Passes" card shows the student their current balance of late passes. Each pass type is clearly listed along with its description and the number of passes remaining (e.g., "24-Hour Extension: 5 available"). This immediate feedback is crucial for helping students make informed decisions about when and how to use their passes.

2. Assignment List: The main body of the interface consists of a list of "Current Assignments." This list is populated by data fetched from the backend, which in turn syncs with the course's assignments in Canvas. Each assignment card displays the title, due date, and a clear "Apply Pass" button for each available pass type. This integrated view ensures students know exactly which assignments are eligible for extensions.

3. Applying a Pass: The process of applying a pass is designed to be a simple, two-click, self-service workflow. When a student clicks a pass to apply to a assignment, a modal will pop up with confirmation details.

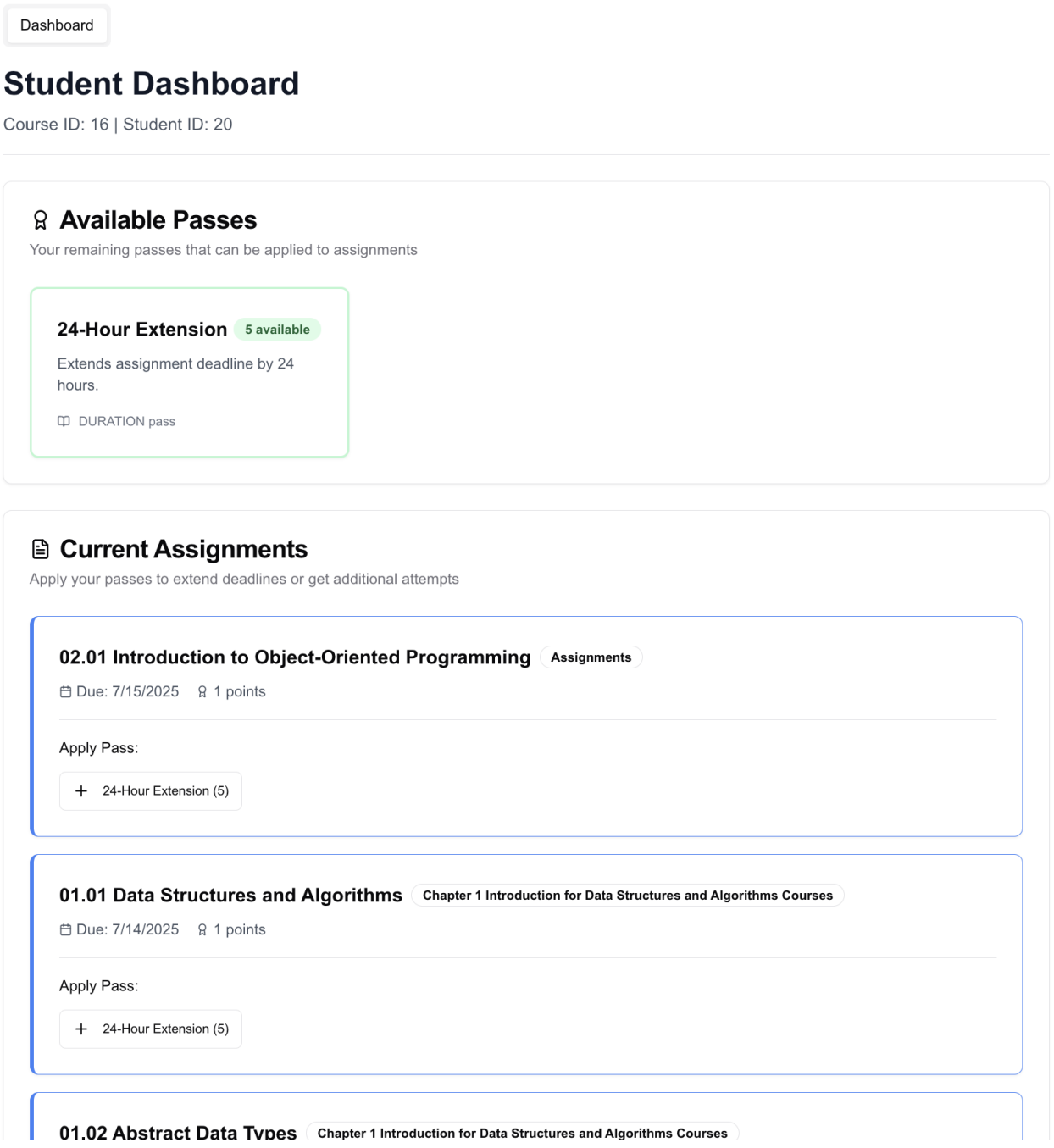


Figure 4.7: Student Dashboard

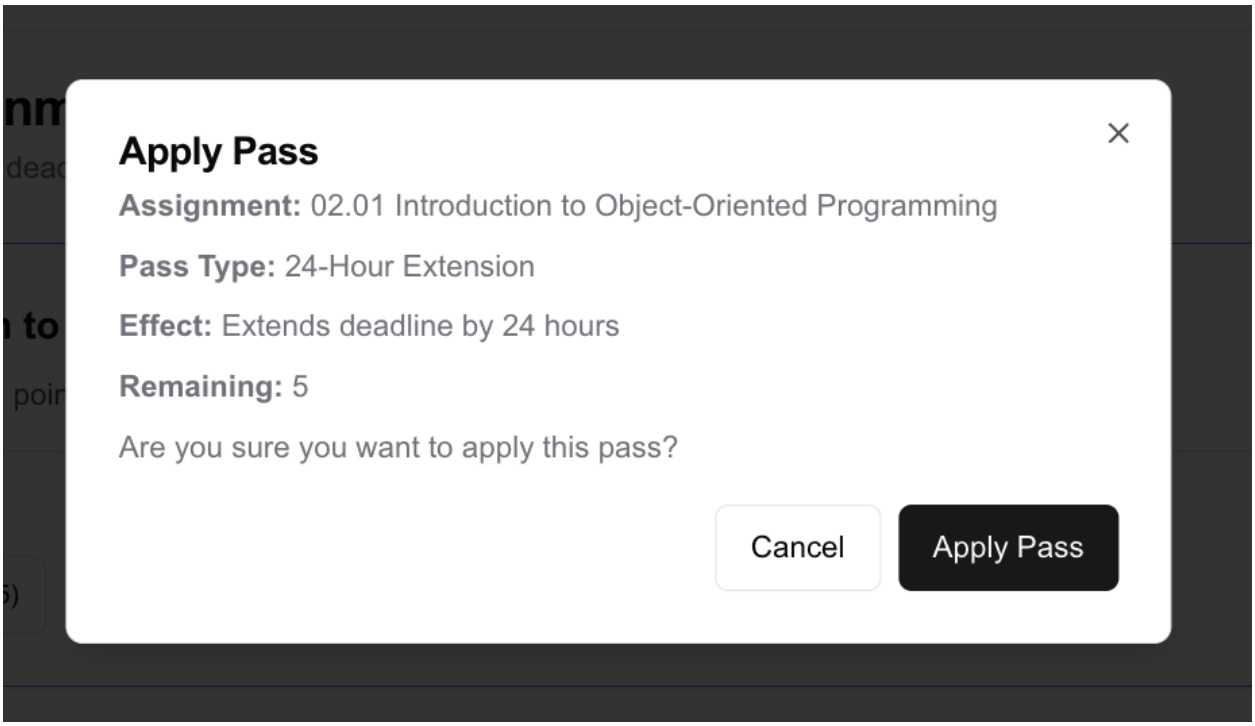


Figure 4.8: Student Pass Application Modal

4. (a) The student clicks a pass that is available to apply, under the desired assignment.

   (b) A confirmation modal (Figure 4.8) appears, summarizing the action: the assignment name, the pass type, its effect (e.g., "Extends deadline by 24 hours"), and the student's remaining pass balance.

   (c) The student clicks the final "Apply Pass" button in the modal to confirm. The frontend then sends the request to the backend, and the UI enters a loading state. Once the backend confirms that the pass has been successfully applied across all systems, the UI updates in real-time to reflect the student's new pass balance and displays a success message.

This streamlined workflow eliminates the need for students to email instructors or teaching assistants, reducing student anxiety and the psychological barriers associated with asking for an extension.

Chapter 5

Traditional Late Policy vs. Flexible Free Pass Policy Study

This chapter details the quasi-experimental methodology used to compare the effects of two different submission policies on student performance and submission behavior. The study analyzes data from two distinct 16-week semesters of CS 1114: Introduction to Software Design at Virginia Tech. As a required course for students intending to major in computer science, CS 1114 is typically taken by first-year students and is delivered in person across multiple sections taught by different instructors.

This investigation compares two cohorts operating under different policy conditions. The first cohort, from Fall 2019, consisted of 673 students across 2 sections and was subject to a traditional, punitive late policy with a 10% grade deduction per day. The second cohort, from Fall 2024, consisted of 421 students across 3 sections and was subject to a flexible resubmission policy through a token-based free-pass system. In this second policy, instructors manually tracked student pass usage, which involved processing Google Forms submissions and updating Canvas, OpenDSA, and Webcat. This semester is crucial to the analysis, as it allows for an examination of how a token-based policy impacts course-level outcomes and student behavior, while also highlighting the significant administrative challenges that motivate the development of the EGP Broker.

By analyzing and contrasting the data from these two semesters, this research directly ad-

dresses the following question:

> In a CS1 course, how does a free pass based system impact student performance and course level outcomes?

To that end, the remainder of this chapter is structured as follows. Section 5.1 will first detail the research context and provide a specific breakdown of the two policy conditions under investigation. Following this, Section 5.2 will outline the data collection and preparation procedures, describing the specific data sources and the steps taken to clean and process the data. Section 5.3 will then present the data analysis plan, specifying the statistical methods used to compare student outcomes. Finally, Section 5.4 will address the ethical considerations of the study.

5.1 Research Context and Policy Conditions

This study was conducted within CS 1114: Introduction to Software Design, a 16-week foundational course at Virginia Tech taught in Java. The course introduces fundamental concepts of object-oriented programming, basic software engineering principles, and simple data structures. The two semesters under investigation shared this core curriculum and content, but differed significantly in their pedagogical approach to assessment and deadlines.

5.1.1 Fall 2019 - Punitive 10% Per Day Deduction Policy

The Fall 2019 cohort consisted of 673 students in 2 sections. The course structure and policies for this semester reflected a traditional, deadline-focused approach. The final grade

was calculated using a percentage-based weighting system, with the components being Participation/Quizzes/In-class Activities (5%), Homework (5%), Programming Assignments (30%), two Midterm Tests (25%), a Final Exam (20%), and Lab Assignments (15%). The policy for late work was inflexible for most low-stakes assignments. For the high-stakes programming assignments, a punitive deduction was applied: a penalty of 10 points (out of 100) was deducted for each 24-hour period an assignment was late, for a maximum of three days.

5.1.2 Fall 2024 - Manual Free Pass Policy

The Fall 2024 cohort consisted of 421 students across 3 sections. This semester marked a significant pedagogical shift toward a mastery-based learning model that prioritized student growth and flexibility. The course adopted a 1000-point cumulative grading system, replacing the traditional midterm exams with eight shorter, more frequent quizzes to support spaced learning and reduce high-stakes pressure. A form of specifications grading was introduced for lab and programming assignments using a four-point EMRN rubric (Excellent, Meets Expectations, Revision Needed, Not Assessable), which encouraged students to improve their work through iterative revision. Central to this model was a flexible "resubmission token" system that replaced punitive late penalties with a structured opportunity for continued improvement. Each student began the semester with six resubmission tokens, which could be redeemed to revise and resubmit assignments or retake a quiz after the original deadline.

Each student began the semester with six resubmission tokens, which could be redeemed to revise and resubmit assignments or retake a quiz after the original deadline. These tokens were administered manually via a Google Form, where students submitted requests that were reviewed and processed by the instructional staff. The policy emphasized learning from feedback by allowing resubmissions only after initial feedback was released. Assignments

became eligible for resubmission during weekly cycles and remained open for up to three cycles (three weeks) post-feedback. For quizzes, a single resubmission was permitted during the week following feedback release. Using a free pass re-opened the assignment or quiz for an additional 24-hour window of submissions and automated feedback. Students were limited to one free pass per week and encouraged to plan their revisions accordingly. If a student exhausted all six passes but still required additional flexibility, they could submit a justification form to request more. While the system aligned well with the course's emphasis on deliberate practice and self-regulated learning, its manual administration required substantial coordination by the instructional team.

5.1.3 Programming Assignments: The Focus of Analysis

The primary focus of the comparative analysis in the following chapter is on the programming assignments, as they represent the most significant, work in the course and were directly affected by the differing policies. In Fall 2019, there were six programming assignments that were collectively worth 30% of the final grade. These were graded on a traditional 100-point scale, and the late penalty was applied directly to this score. A grade of zero on any programming assignment could result in failing the course, regardless of the student's overall average. In Fall 2024, there were also six programming assignments, each worth 50 points on the 1000-point scale (for a total of 300 points, or 30% of the final grade). Students could use one of their six resubmission tokens to gain a 24-hour window to resubmit a programming assignment after an initial round of grading was complete. This allowed them to improve their score from an "R" (Revision Needed) to an "M" (Meets Expectations) to receive full credit. By focusing on the scores and submission patterns for these assignments, this study can directly compare the impact of a punitive deduction versus a flexible, resubmission token-based system.

5.2 Data Collection and Preparation

To construct a comprehensive dataset for this analysis, information was collected and integrated from two primary systems for both the Fall 2019 and Fall 2024 semesters. The course's Learning Management System, Canvas, served as the source for all high-level student outcome data. From Canvas, final course rosters, final letter and numeric grades, withdrawal records, and a binary indicator of whether each student attempted the final exam were exported. This information was essential for establishing baseline comparisons of the two cohorts and for categorizing students into performance groups for more granular analysis.

Complementing the course-level data from Canvas, all submission-level data for the six programming assignments was obtained from Web-CAT [10], the automated grading system used in the course. The Web-CAT logs provided a detailed record of every submission made by every student for every assignment, including timestamps and performance metrics. This granular dataset was critical for analyzing student submission behaviors and the quality of their work.

From these two sources, several key variables were extracted for the analysis. To assess overall course performance, the student's final grade was collected in both letter form (Final Grade) and as a numeric GPA (Final_Grade_Numeric). Student persistence was measured using a binary variable, Attempted Final, indicating whether a student completed the final exam. To measure the quality of work on individual programming assignments, the Ref Test % was used. This variable represents the percentage of reference tests passed by a student's final submission, providing an objective and unbiased measure of correctness. Finally, to track student effort and work patterns, the Submission Number was recorded for each attempt on an assignment.

5.2.1 Data Cleaning and Anonymization

Prior to analysis, the raw data from both Canvas and Web-CAT underwent a critical cleaning and anonymization process to ensure student privacy and data integrity. All personally identifiable information, including student names, email addresses, and university ID numbers, was removed from the dataset. Each student was assigned a unique, randomly generated numerical identifier. This same identifier was used for a given student across all datasets (Canvas grades, Web-CAT submissions, etc.) to enable the merging of their records for a holistic analysis, while making it impossible to trace the data back to an individual. Once this process was complete, the original, identifiable data was discarded from the research dataset. Furthermore, all records for students who officially withdrew from the course were filtered out of the final dataset to ensure that the analysis focused only on students who completed the semester.

5.3 Data Analysis Plan

The data analysis was designed to proceed in a series of logical steps, moving from a high-level comparison of the two cohorts to a more granular investigation of student behavior and performance. This approach was chosen to first establish the overall context and then to systematically drill down into the nuanced effects of the different late work policies.

The analysis begins with the use of descriptive statistics to compare high-level course outcomes, such as withdrawal rate, pass rate, average final exam grade, average final course grade. For this, Cohen's d is used to quantify the difference between the means and ANOVA is used to determine statistical significance. This initial step is crucial for establishing a baseline and understanding whether the two cohorts, despite differences in size and struc-

ture, had comparable overall success, which contextualizes any subsequent findings about submission behavior.

Next, the analysis focuses on quantifying how students engaged with the flexibility offered by each policy. This includes metrics such as the percentage of late submissions in Fall 2019 and the proportion of students who resubmitted work at least once in Fall 2024. While the two policies serve different purposes-Fall 2019 enforced a late submission penalty (10% deduction per day) whereas Fall 2024 encouraged revision through resubmissions-placing them side by side highlights how students made use of the options available. In this way, Fall 2019 provides a baseline for understanding patterns of engagement, allowing us to see how student behavior shifted when the form of flexibility changed.

To investigate how different types of students responded to the flexibility provided in Fall 2024, the cohorts were segmented into three performance groups based on their final course grades: High Performers (A- to A), Good Performers (C to B+), and Struggling Students (F to C-). Fall 2019 serves as a useful baseline for this comparison, as it reflects how the same categories of students behaved when flexibility was defined instead by a late submission penalty. This segmentation allows us to see not only overall usage patterns, but also whether students at different achievement levels engaged with policy-driven flexibility in distinct ways.

The performance analysis then examines the relationship between policy use and the quality of student work, measured by the reference test percentage score. Here, an ANOVA was used to evaluate whether these differences are statistically significant and Cohen's d to message the difference in means, both for the overall cohort and within each performance group. To capture patterns that simple averages might obscure, score distributions are visualized using histograms-particularly to highlight outcomes such as increases in different reference test percentage score under the late policy or improvements in performance after resubmission

opportunities.

5.4 Results of the Comparative Policy Study

This section presents the quantitative findings from the comparative analysis of the two late work policies implemented in the Fall 2019 (punitive deduction) and Fall 2024 (manual token-based) semesters of CS 1114. The analysis follows the methodology outlined in Section 5.3 and seeks to answer the primary research question: in a CS1 course, how does a free pass based system impact student performance and course level outcomes? The results are presented and discussed in a progressive manner, beginning with high-level course outcomes and drilling down into more granular analyses of student behavior and performance of Programs, across different achievement levels (defined in Section 5.3).

5.4.1 High-Level Course Outcomes

To establish a baseline for comparison, this section presents the top-level statistics related to student retention and course-level outcomes. The data, summarized in Table 5.1, reveals several notable differences between the semester with the traditional rigid punitive late policy (Fall 2019) and the semester with the flexible resubmission token policy (Fall 2024).

Observing the Table 5.1, a key difference is in student retention. The Fall 2024 cohort, which operated under the flexible token-based policy, had a substantially lower withdrawal rate (13.39%) compared to the Fall 2019 cohort's 21.68%. This suggests that the flexibility offered by the late pass system, which was part of a broader pedagogical shift toward the ability to resubmit, may have played a role in helping more students persist in the course

Table 5.1: Course level outcomes for Fall 2019 and Fall 2024

Metric	Fall 2019	Fall 2024	Cohen's d	p-Value
Withdrawals	21.68%	13.39%	0.14	< 0.005
Took Final Exam	75.73%	83.07%	0.11	< 0.005
Course Pass Rate ($\geq$ C)	61.37%	68.11%	0.22	< 0.005
Final Exam Avg. Score	61.99%	66.94%	0.27	< 0.001
Final Grade Avg.	C+ to B- (2.541)	B- to B (2.813)	0.27	< 0.002

* Pass rate defined as students earning a final grade of C or higher.

rather than withdrawing when they fell behind early on or felt that they had dug themselves into a hole they could not get out later in the semester.

This trend is further supported by the final pass rate. In Fall 2024, 68.11% of the students successfully completed the course with a GPA of 2.0 or higher, compared to 61.37% in Fall 2019. Correspondingly, a higher percentage of students in the Fall 2024 semester sat for the final exam (83.07%) than in the Fall 2019 semester (75.73%). Taken together, these high-level metrics indicate that the semester with the flexible, token-based resubmission policy was associated with improved student retention and a higher overall rate of success.

Beyond retention and pass rates, there were also measurable gains in performance outcomes. The average final exam score increased from 61.99% in Fall 2019 to 66.94% in Fall 2024 ($p < 0.001$), representing a small but meaningful effect size. Similarly, average course grades shifted upward from the C+/B- range (2.541) to the B-/B range (2.813), again with a small but significant effect ($p < 0.002$). These improvements in performance suggest that the policy not only influenced persistence but also enhanced learning outcomes. The flexibility potentially reduced stress and allowed students to focus on mastery rather than rigid deadlines.

The observed differences, though modest in absolute magnitude, are consistent across multiple metrics (withdrawals, exam participation, pass rates, exam performance, and final

grades). The convergence of these results drives the conclusion that the token-based system provided a more supportive environment that enabled a broader range of students to remain engaged and succeed. While the effect sizes are small by conventional thresholds, the practical significance in an educational context (fewer withdrawals, higher persistence, and stronger performance) points toward meaningful benefits for student learning and retention.

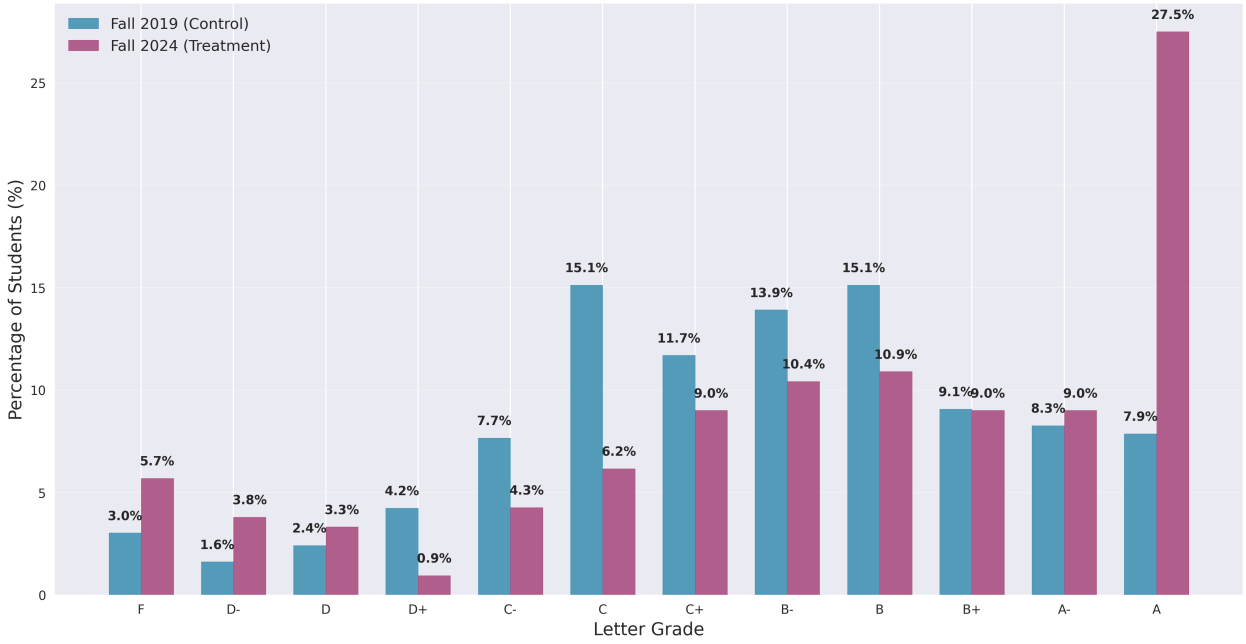


Figure 5.1: Final Grade Distribution of Students Who Attempted Final Exam

Looking at Figure 5.1 distribution of final letter grades reveals a notable shift between the two cohorts. In Fall 2019, student performance was most heavily concentrated in the C to B range, with relatively few students achieving an A. By contrast, in Fall 2024 there was a marked increase in the proportion of students earning an A (27.5% compared to just 7.9% in 2019). The B+ to A- range remained relatively stable across semesters, but the proportion of students in the D+ to B range declined substantially in 2024 (6.2% versus 15.1% in 2019). At the lower end of the distribution, the percentage of students receiving grades in the F to D range increased slightly.

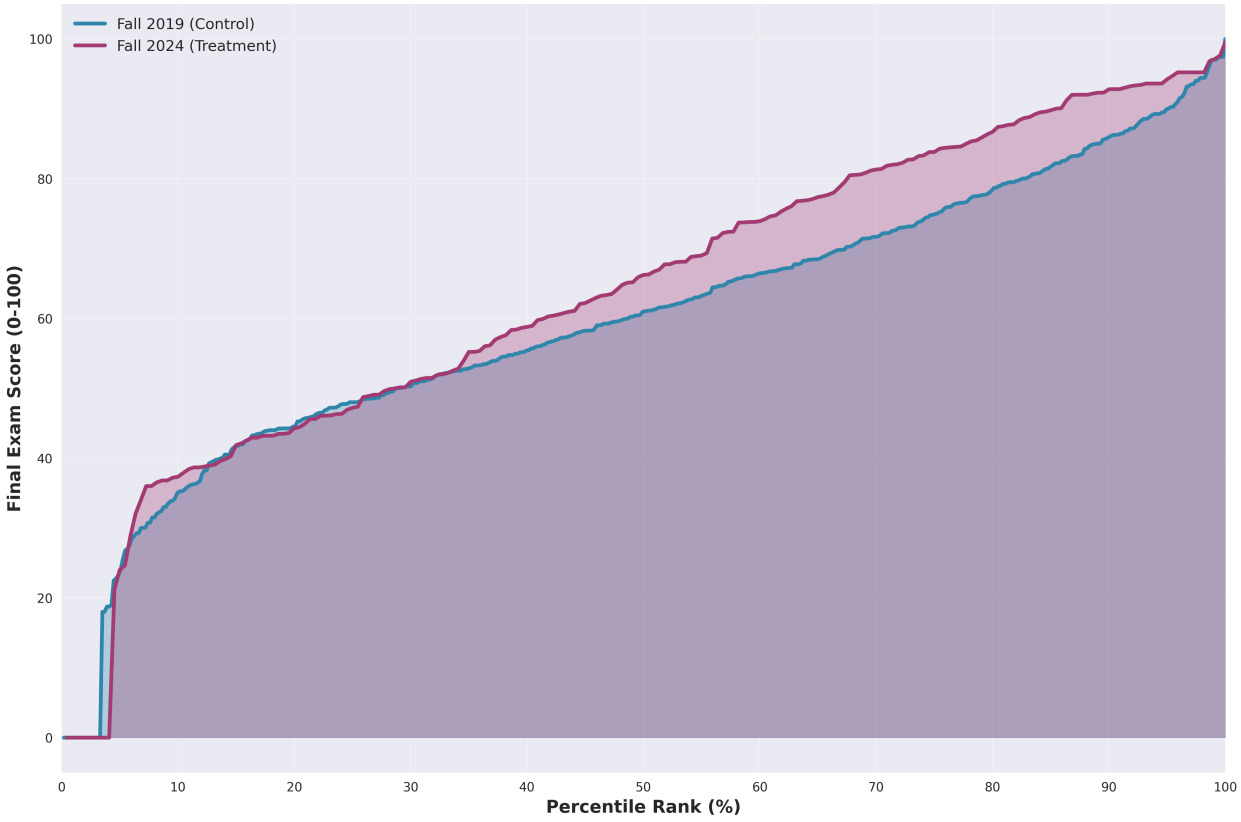


Figure 5.2: Final Exam Score vs. Percentile

Examining the cumulative distributions of final exam scores in Figure 5.2, several patterns stand out. At the very low end of the distribution (bottom 5–10% of students), both cohorts follow nearly identical trajectories, with a cluster of students scoring near zero. This indicates that the token-based resubmission policy did not substantially alter outcomes for students who were possibly disengaged in the course. However, beginning around the 15th percentile, the Fall 2024 (treatment) curve begins to pull ahead of the Fall 2019 (control) curve. This gap gradually widens through the middle of the distribution, suggesting that the policy was most impactful for students in the broad middle tier.

The divergence continues to grow steadily, reaching its largest magnitude between roughly the 60th and 85th percentiles, where the Fall 2024 course shows several points of advantage in the final exam scores. This implies that the additional flexibility provided by the token-based system may have been particularly beneficial for moderately strong students, allowing them to consolidate their understanding and achieve higher levels of performance. Even at the very top of the distribution (90th percentile and above), the Fall 2024 cohort maintains a slight but consistent edge, culminating in a higher ceiling for the strongest performers.

Taken together, these results indicate that the observed benefits of the policy were not isolated to a narrow band of students but distributed across the majority of the achievement spectrum. Rather than dramatically lifting only the lowest or highest performers, the policy appears to have elevated performance more evenly, with its strongest relative gains concentrated among middle achievers. This pattern is constant the earlier findings on overall exam averages and final grades, reinforcing the conclusion that the token-based system supported a broader range of students in achieving higher mastery and stronger final exam outcomes.

Taken together, these shifts suggest that many of the “middle” students who clustered in the C to B range in 2019 appear to have moved upward into the B+ to A range by 2024. The result is a distribution that is no longer centered around average performance but instead

shows a pronounced concentration at the top end, with a much larger share of students achieving the highest grade. This upward shift aligns with the broader trends observed in exam scores and pass rates, and it is consistent with the idea that the token-based late work policy helped support students in ways that enabled them to reach higher levels of achievement rather than plateauing in the middle of the grade scale.

5.4.2 Overall Policy Utilization

Beyond course-level outcomes, the data reveals a shift in how students engaged with the respective policies. The analysis shows a dramatic increase in both the volume of taking advantage of the resubmission and the breadth of student adoption. This suggests that the shift from a punitive policy to a flexible token-based resubmission system significantly altered students' behavior.

Table 5.2: Policy Engagement in Fall 2024 and Fall 2019

Metric	Fall 2019	Fall 2024	Cohen's d	p-value
% of All Submissions that used Policy	17.6%	31.8%	0.348	< 0.001
% of Students Using Policy ≥ 1 Time	52.5%	81.0%	0.606	< 0.001
Avg. Uses (among all students)	1.06	1.91	0.643	< 0.001
Avg. Uses (among users of policy)	2.01	2.36	0.302	< 0.002

As shown in Table 5.2, the proportion of all submissions utilizing the policy nearly doubled, rising from 17.6% in Fall 2019 to 31.8% in Fall 2024. Similarly, the proportion of students using the policy at least once increased from 52.5% to 81.0%. Examining the average number of uses provides further insight. Across all students, the average increased from 1.06 to 1.91, reflecting the broader adoption of the policy. Among students who actively used the policy, the average increased from 2.01 to 2.36, a larger relative increase than the overall mean. This

difference is important, it indicates that not only did more students take advantage of the policy, but those who used it engaged with it more frequently, suggesting deeper integration of the flexible system into their workflow and strategic planning. In other words, the shift was not only in breadth (more students participating) but also in depth (students leveraging the policy multiple times), highlighting a substantial change in behavior under the token-based system. These changes, supported by statistically significant differences (Cohen's d ranging from 0.302 to 0.643, all $p < 0.01$), demonstrate a meaningful behavioral shift toward actively managing deadlines using flexible submission options.

This change in perception is further evidenced by the massive increase in students exercising the free pass policy. In Fall 2019, only about half the class (52.5%) ever submitted a late assignment. In contrast, the vast majority of students (81.0%) in Fall 2024 used at least one late pass. This indicates that the token-based system was perceived not as a last resort for students who were behind, but as a universal tool for managing workload and prioritizing mastery, leading to its adoption as a near-standard practice.

Table 5.3: Distribution of Policy Uses on Programs

# of Programs	% of Students (Fall 2019)	% of Students (Fall 2024)	Cohen's d	p-value
0	47.5%	19.0%	0.606	< 0.001
1	23.2%	24.2%	0.022	< 0.800
2	12.9%	22.7%	0.270	< 0.001
3	10.1%	19.0%	0.267	< 0.001
4+	6.3%	15.2%	0.315	< 0.001

The distribution of late submissions per student, visualized in Figure 5.3 and Table 5.3, reveals the depth of this behavioral change. In Fall 2019, nearly half of the students (47.5%) never submitted an assignment late, suggesting a strong adherence to deadlines when faced with a penalty. This figure plummeted to just 19.0% in Fall 2024. This shift away from

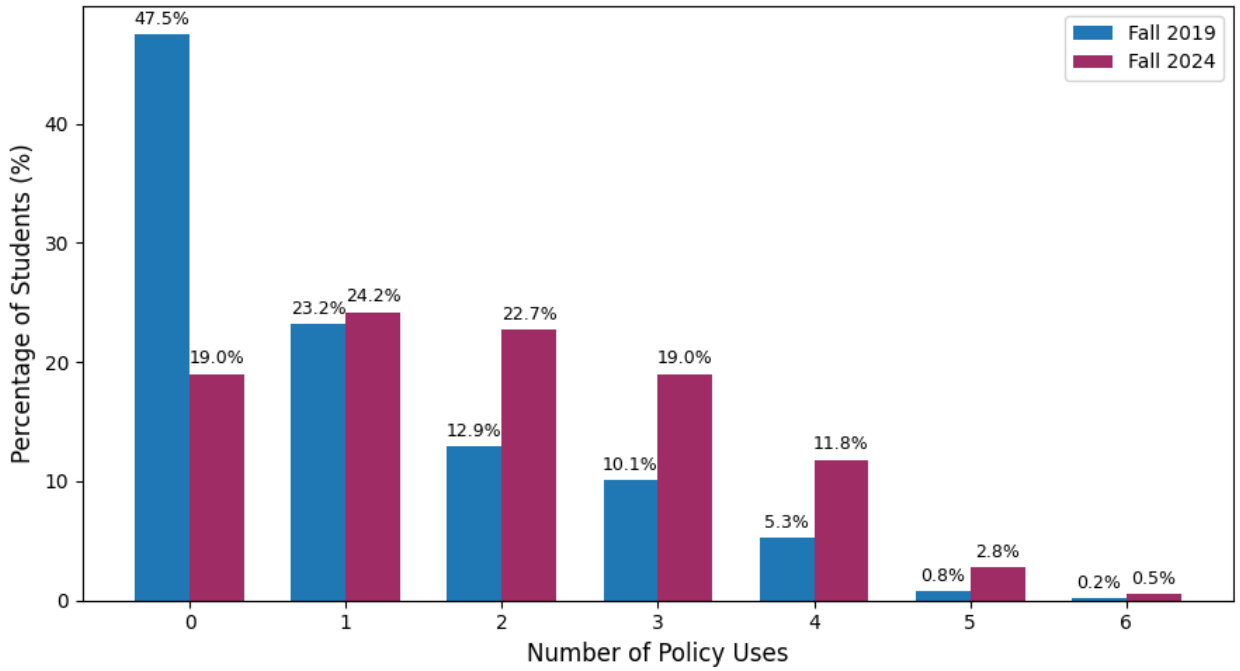


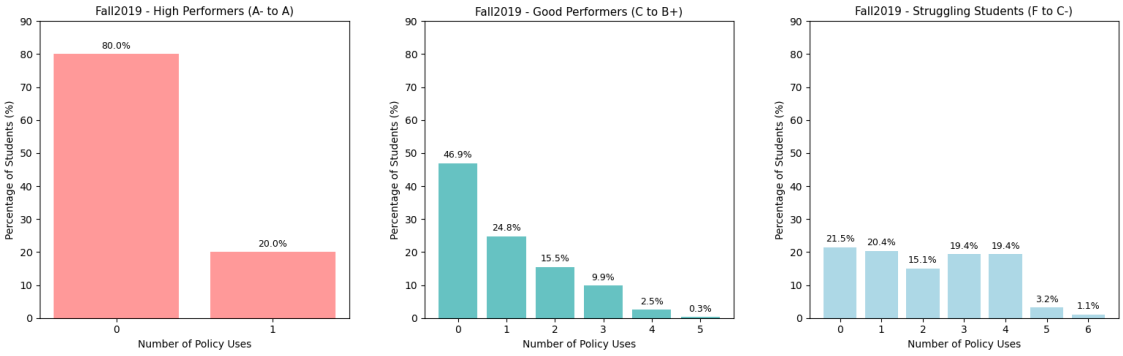
Figure 5.3: Distribution of Policy Use Frequency Among Students (Fall 2019 vs Fall 2024)

strict adherence suggests that students in the token-based system may have felt empowered to use the flexibility offered to them. The most notable change is in the proportion of "heavy users" (those with 4 or more late submissions), which more than doubled from 6.3% in Fall 2019 to 15.1% in Fall 2024. This increase in frequent use is consistent with the mastery-based learning goals of the Fall 2024 course design; students were encouraged to revise and resubmit work, and the token system provided the necessary mechanism for them to take the extra time needed to do so, rather than potentially settling for a lower grade on a rushed submission.

5.4.3 Policy Utilization by Performance Group

To investigate whether the flexible policy had a differential impact, the analysis was segmented by final course GPA. These performance groups were not chosen arbitrarily; rather,

they reflect meaningful academic thresholds for the student population. The Struggling Students (0.0-1.7) group, representing grades from F to C-, includes those who are failing the course, as a GPA below 2.0 is the minimum required for computer science majors and minors to continue in their program's coursework. The Good Performers (2.0-3.3) group, representing grades from C to B+, represents the broad middle range of students who are successfully meeting the requirements to progress in their major. Finally, the High Performers (3.7-4.0) group, representing grades from A- to A, represents students who are excelling in the course. To ensure that the conclusions drawn from this segmented analysis are robust, statistical tests were used to validate the observed differences.



(a) A- to A Students (b) C to B+ Students (c) F to C- Students

Figure 5.4: Policy Usage by GPA Category Across Fall 2019

(a) A- to A Students (b) C to B+ Students (c) F to C- Students

Figure 5.5: Policy Usage by GPA Category Across Fall 2024

Table 5.4: Policy Usage By Grade Category

Performance Group	Fall 2019	Fall 2024	Cohen's d	p-value
High Performers (A- to A)	3.30%	18.60%	0.507	< 0.00001
Good Performers (C to B+)	16.20%	33.90%	0.447	< 0.00001
Struggling Students (F to C-)	34.80%	53.50%	0.388	< 0.00001

The distributions in Figure 5.4 and Figure 5.5, along with the tabular summary in Table 5.4, provide a detailed view of behavioral shifts within each performance group. For High Performers, the change is stark: in Fall 2019, 80% of this group never submitted an assignment late. In Fall 2024, this number dropped to just 32.5%, with the majority of students using one or two passes. For Good Performers, the distribution in Fall 2019 was heavily skewed towards zero or one late submission; in Fall 2024, it became much more uniform, with a significant increase in students using two or three passes. The distribution for Struggling Students was already spread out in Fall 2019, and it remained so in Fall 2024, though with a noticeable drop in the percentage of students who never submitted late. However, looking deeper at the distribution of number of late submissions used by Struggling students, we see usage rates are similar for zero and one late submissions, but there is a strong peak at two late submissions, where the usage percentage doubled (31.6% in Fall 2024 and 15.1% in Fall 2019), and drastically decreased from 19.4% to 7.9% for 3 late submissions. This shows a trend of a lower percentage of struggling students submitting late for over half the programs. This shows a positive trend of less students falling into sustained procrastination compared to Fall 2019.

The data in Table 5.5 reveals a striking and divergent pattern of behavior across the performance groups. A Chi-square test for independence confirmed that the relationship between a student's GPA category and their late submission behavior is statistically significant ($p < 0.001$). This validates the observation that students in different performance groups interact

Table 5.5: Policy utilization by performance group for Fall 2019 (punitive) and Fall 2024 (token-based) policies.

Metric	Performance Group	Fall 2019	Fall 2024	Cohen's d	p-value
% of Students	High Performers	20.0%	67.5%	1.086	< 0.001
Using Policy $\geq$	Good Performers	53.1%	86.5%	0.711	< 0.001
1 Time	Struggling Students	78.5%	94.7%	0.440	0.024
Avg. Late	High Performers	0.20	1.12	1.161	< 0.001
Submissions	Good Performers	0.97	2.03	0.908	< 0.001
(All Students)	Struggling Students	2.09	3.21	0.733	< 0.001

with late policies in fundamentally different ways. The shift to a flexible, token-based policy had the most dramatic effect on High Performers. Under the per day deduction policy of Fall 2019, only 20% of these students ever submitted late. The direct grade penalty was likely perceived as an unacceptable risk to their high standing. In Fall 2024, their behavior changed entirely; with the penalty removed, 67.5% used at least one pass, and their average usage increased by a remarkable 450%. An independent sample t-test confirmed this change was statistically significant ($p < 0.0001$), meaning there is a less than 0.01% probability that this shift occurred by random chance. This provides strong evidence that high-performing students adopted the token system as a strategic tool for managing deadlines without fear of penalty.

Good Performers, representing the core of the class, also embraced the flexible policy, though less dramatically. Their adoption of the policy increased by 51%, and their average usage rose by 69%. This change was also found to be statistically significant ($p < 0.001$), indicating that this large middle group, who may have been hesitant to accept a grade penalty, found the token system to be a much more accessible and useful tool for managing the normal pressures of the course.

Most surprisingly, the behavior of Struggling Students moved in the opposite direction. In

Fall 2019, this was the group that submitted late most often, with 78.5% submitting late at least once for an average of 2.09 late submissions per student. This high usage suggests that they were frequently falling behind and were forced to accept the grade penalty as a recurring consequence. However, with the introduction of the token-based system in Fall 2024, their average usage decreased significantly, a change that was also statistically significant ($p = 0.0053$). This counterintuitive finding suggests two possibilities. First, the structure of the token system, which provided a finite and visible resource, may have encouraged better time management and self-regulation compared to the punitive system. Instead of learning time management or addressing the root causes of their stress, some students might be regularly submitting late work because they feel overwhelmed, anxious, or unable to keep up. Alternatively, it is possible that these students faced systemic challenges so significant that even the flexible policy could not fully mitigate them, a possibility that will be explored further in the analysis of assignment scores.

This detailed analysis of submission behavior has established that students in different performance groups used the policies in fundamentally different ways. However, it does not yet answer the critical question of impact: did using these policies actually affect the quality of student work? For the High Performers who used the token system strategically, did taking extra time result in higher-quality submissions? Conversely, for the Struggling Students who frequently submitted late under the punitive policy, what was the effect on their scores? To answer these questions, the next section will analyze the reference test scores for programming assignments.

5.4.4 Impact on Assignment Performance (Reference Test Scores)

While the previous section detailed how students used the late policies, this section investigates the critical question of impact: did using these policies affect the quality of student work? To answer this, the analysis focuses on the reference test percentage score for programming assignments. This metric was chosen as the primary measure of assignment quality because it provides a direct, objective assessment of a program's correctness. The reference tests are a standardized suite of automated tests that are applied to every student's submission, ensuring a consistent and unbiased evaluation that is comparable across both semesters. This is in contrast to the final assignment grade, which can include subjective components, such as scores for design and style, that are graded by instructors or TAs and may vary.

Table 5.6: Average reference test percentage by performance group and semester

Performance Group	Fall 2019	Fall 2024	Cohen's d	p-value
High Performers (A- to A)	97.8%	97.5%	0.032	0.628
Good Performers (C to B+)	93.3%	87.0%	0.335	< 0.001
Struggling Students (F to C-)	81.3%	58.8%	0.651	< 0.001

The data, from Figure 5.6, reveals a statistically significant decrease in average reference test performance between the two semesters, from 92% in Fall 2019 to 86% in Fall 2024 ($t = 7.84$, $p < 0.0001$). However, this aggregate number masks a highly nuanced story. When segmented by performance group, as shown in Table 5.6, it becomes clear that the negative impact was not evenly distributed throughout the student population.

The performance of High Performers was remarkably resilient. Despite their dramatic increase in late submissions, their average reference test score remained unchanged at 98%. A t-test confirmed that this small difference was not statistically significant ($p = 0.6277$). This provides strong evidence that this group used the flexible policy strategically and effectively,

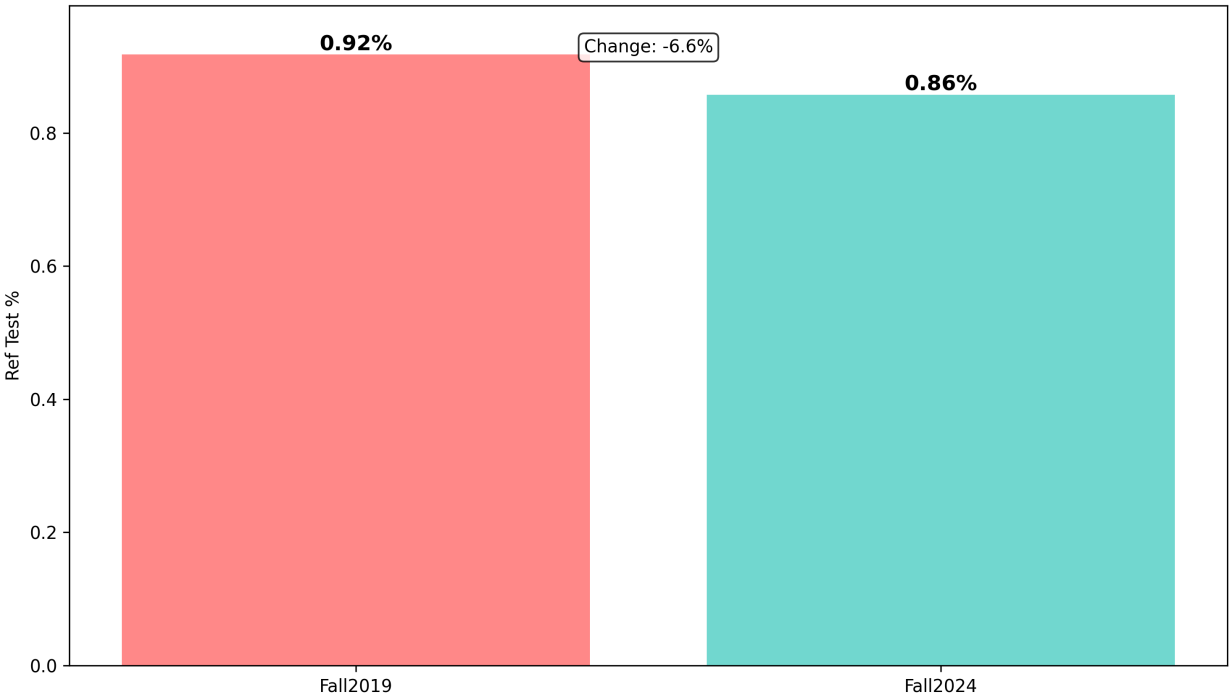


Figure 5.6: Overall Average Ref Test % Comparison

taking extra time when needed without any detriment to the quality of their work.

For Good Performers, the shift to the token-based policy was associated with a moderate but statistically significant decline in performance, with their average score dropping from 93% to 87% ($p < 0.0001$). This suggests that for this middle group, the increased flexibility may have introduced a trade-off between timeliness and quality, leading to a slight decrease in overall performance on programming assignments.

The most alarming finding is the severe decline in performance among Struggling Students. This group's average reference test score plummeted by 22 percentage points, from 81.0% in Fall 2019 to just 59.0% in Fall 2024. This decline was highly statistically significant ($p < 0.0001$). While the average scores tell part of the story, the distribution of scores, shown in Figure 5.2, provides a much clearer picture of the problem.

The box plot for Struggling Students in Fall 2024 (Figure 5.7) is starkly different from its

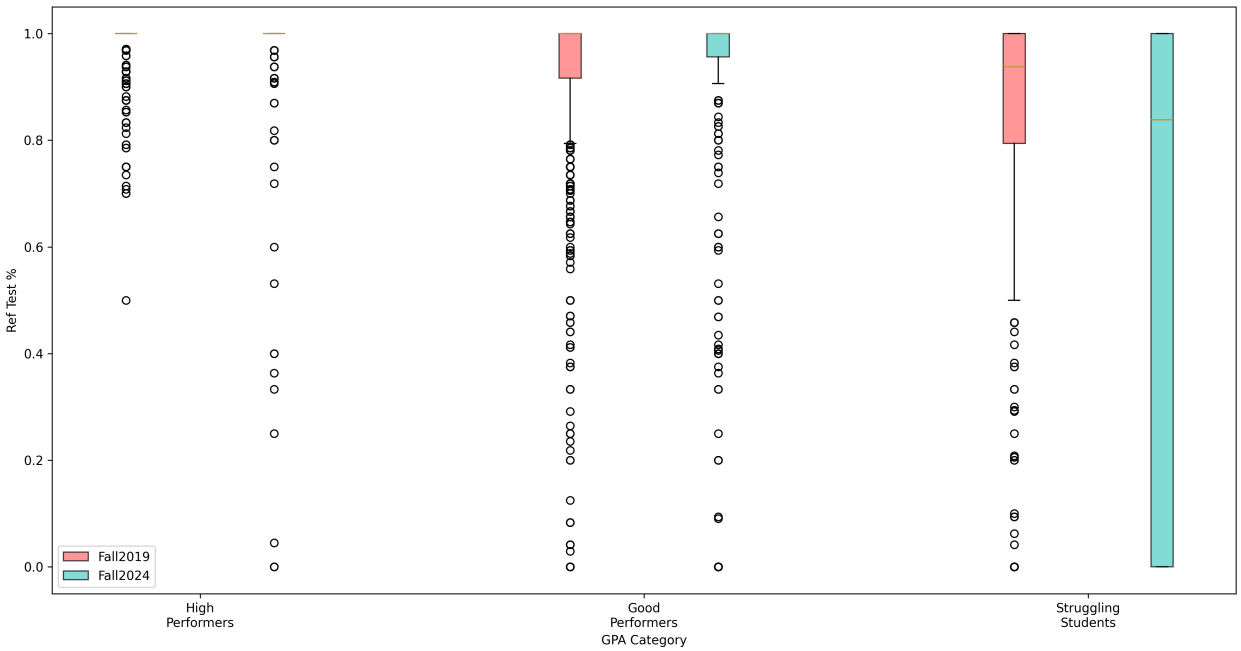


Figure 5.7: Box Plot Showing Distribution of Ref Test % by GPA Group

2019 counterpart. The median score is visibly lower, and the entire interquartile range has shifted downwards. Most critically, the first quartile (Q1) of the distribution is at 0%. This confirms the quartile analysis finding and visually demonstrates that a large number of submissions from this group in Fall 2024 received very low or zero scores.

This analysis of average performance reveals a positive trend for High and Good preforming students, as there is a higher concentration of students above the 80% Reference Test % mark. However, the average scores decreased and for the most vulnerable students, there is much higher concentration of students receiving zeros. This does not fully explain the relationship between using the late policy and the quality of the submitted work. Did taking extra time help struggling students, even if their overall average was lower? And what explains the dramatic increase in extremely low score submissions for this group in the flexible semester? Is there a large percentage of zero/no submission scores for Fall 2024 or are they low scores on attempts? The next section will delve deeper into these questions by directly comparing

the scores of on-time and late submissions.

5.4.5 Investigating the Decline in Performance for Struggling Students

The previous section highlighted a severe decline in the average reference test scores for Struggling Students in the Fall 2024 semester. To understand the root cause of this trend, this section provides a more detailed investigation into the score distributions for this specific group, with a focus on the alarming increase in zero-score submissions.

Table 5.7: Zero-score submissions for struggling students in Fall 2019 (punitive) and Fall 2024 (token-based) policies.

Metric	Fall 2019	Fall 2024	Cohen's d	p-value
% of Submissions with Zero Score	7.9%	30.3%	0.665	< 0.001
% of Students with ≥ 1 Zero Score	25.8%	86.8%	1.473	< 0.001
Avg. Zero Scores per Student	0.47	1.82	1.225	< 0.001

Table 5.7 reveals a crisis of non-engagement for struggling students under the flexible policy. The percentage of submissions from this group that received a zero score increased nearly fourfold, from 7.9% in Fall 2019 to 30.3% in Fall 2024. This issue was widespread, affecting the vast majority of these students: in Fall 2024, 86.8% of struggling students had at least one zero-score submission, compared to only about a quarter of them in Fall 2019.

A critical discovery from the data is that this increase was not due to students submitting more work that failed all tests, but rather due to students not submitting any work at all. In Fall 2024, 95.7% of the zero scores from struggling students were the result of a non-submission, up from 88.6% in Fall 2019. While the number of students who submitted work that failed all tests remained small and relatively constant (4 students in 2019, 3 in 2024), the number of struggling students with at least one non-submission skyrocketed from 21 to

31, affecting 81.6% of the entire group.

This presents a policy paradox. The flexible, token-based system, which was intended to reduce stress and provide a safety net, appears to have inadvertently created a pathway for disengagement for the most vulnerable students. However, it is crucial to note that this comparison is confounded by another course policy specific to the Fall 2019 semester. The syllabus for that semester included a "minimum work requirement" which stated: "A grade of zero on any program assignment may result in a grade of F in the course, regardless of your actual overall numeric score, at the discretion of the instructor." This high-stakes policy, which was unrelated to the late policy, created a powerful incentive for students to submit something for every assignment, even if it was incomplete, to avoid the risk of automatic failure. The absence of this policy in Fall 2024 means that the two semesters are not directly comparable on the metric of non-submissions alone. The low rate of non-submissions in Fall 2019 is likely a reflection of this separate, high-stakes requirement rather than an effect of the punitive late policy itself.

Despite the drastic increase in zero-score submissions among struggling students in Fall 2024, it is critical to contextualize this finding within the broader picture of student success. Even without the high-stakes "minimum work requirement" policy of 2019, the overall percentage of the class that fell into the "Struggling Students" category still saw a slight improvement, decreasing from 18.8% in Fall 2019 to 18.0% in Fall 2024. This suggests that while the flexible policy may have led to more non-submissions among the students who were already struggling, it did not result in a larger proportion of the overall class falling into this at-risk category.

5.5 Discussion

The comparative analysis of the punitive deduction and flexible token-based late work policies reveals a complex and multifaceted story. The data demonstrates that the shift to a flexible, token-based system in Fall 2024 was associated with clear benefits at the course level, including significantly higher retention and a greater overall pass rate. However, a deeper analysis of student behavior and performance uncovers a more nuanced reality, showing that the effects of this policy change were not uniform and, in some cases, were profoundly different for students with varying levels of academic achievement.

The flexible policy was overwhelmingly embraced by the student body, leading to a dramatic increase in the use of extensions. This behavioral shift was most pronounced among High-Performing students, who adopted the token system as a strategic tool to manage their workload, using extensions far more frequently than under the punitive system without any decline in the quality of their work. This suggests that for students who are already well-equipped with self-regulation skills, a flexible policy can be a powerful and effective tool.

However, the findings for Struggling Students present a significant and cautionary tale. While the flexible policy was intended to provide a safety net, it was correlated with a slight decline in this group's program assignment performance and a crisis of non-engagement, as evidenced by a nearly fourfold increase in zero-score submissions. The data suggests that for students who are already behind, the removal of the structure and urgency imposed by a punitive deadline may inadvertently create a pathway to disengagement. Although this finding is partially confounded by a separate "minimum work requirement" policy in the 2019 semester, the trend is still important to highlight. A major pitfall of the free pass design is that although it is designed to allow students to extend deadlines to allow for additional submissions, it leaves the door open for students prone to procrastination to continually use

the passes to extend the deadline until max period and potentially forget to complete the program.

Ultimately, this analysis demonstrates that there is no one-size-fits-all solution for late work policies. A policy that empowers one group of students may fail to support, or even negatively impact, another. The manual Fall 2024 system requires students to fill out a form and wait for batch processing, may have also contributed to these challenges. An automated system like the EGP Broker, as outlined in Chapter 3, would address many of these issues. It is clear that for instructors, it would eliminate the administrative workload of tracking requests and updating deadlines. However, for students, it would provide immediate, on-demand access to extensions and a clear, real-time dashboard of their pass usage, which could foster better self-regulation. A crucial next step for future research will be to deploy the EGP Broker and analyze its usage data. Comparing a truly low-friction, automated token system against the manual system of 2024 and the punitive system of 2019 will provide a much deeper understanding of how the implementation of a free pass policy affects student outcomes.

Chapter 6

Conclusion and Future Work

This thesis presented a dual-pronged investigation into the role of flexible, token-based late work policies in introductory computer science education. It addressed two interconnected problems: the practical, administrative challenge of implementing such policies at scale, and the need for a deeper, comparative understanding of their impact on student behavior and performance. To that end, this work made two primary contributions: (1) the design and implementation of the EGP Broker and Protocol, an automated system for managing late passes across multiple platforms, and (2) a quasi-experimental data analysis comparing a manually-administered token policy to a traditional punitive deduction policy. This final chapter will summarize the key conclusions drawn from each of these contributions, discuss the limitations of the study, and outline promising directions for future research.

6.1 Conclusions from the EGP Broker and Protocol

The literature and the data from this study both confirm that the primary barrier to the widespread adoption of pedagogically valuable token-based late pass systems is the significant administrative overhead required to manage them manually. The EGP Broker was designed to solve this problem directly. By providing a centralized and automated tool that integrates seamlessly with the LMS, it eliminates the cumbersome and error-prone work of tracking pass usage, processing requests, and updating deadlines.

Furthermore, the EGP Protocol addresses a critical gap in existing solutions by creating a standardized, platform-agnostic method for communicating extensions to any compliant third-party assignment platform. This makes the EGP Broker not just a tool, but an extensible framework that can support the diverse ecosystem of educational technologies used in modern CS courses. The primary contribution of this technical work is a practical, scalable solution that removes the most significant obstacle to implementing flexible and equitable late policies, making them a feasible option for instructors in any course, regardless of enrollment size.

6.2 Conclusions from the Data Analysis

The comparative analysis of the Fall 2019 (punitive) and Fall 2024 (manual token-based) semesters revealed a complex and multifaceted story about the impact of late work policies. At a high level, the flexible, token-based policy showed strong potential to support student success, with clear indicators of improvement in overall pass rates and student retention. The structure gave students more agency over their time, and for many, especially high-performing students (more students became high performers compared to the punitive policy), it became a powerful tool for balancing workload without compromising the quality of their submissions.

However, a more detailed analysis revealed that these benefits were not uniformly experienced across all student groups. While the policy encouraged self-regulation and workload management for some, others (particularly struggling students) appeared to disengage, as evidenced by a sharp rise in zero-score submissions. This suggests that flexibility alone may not be enough to support students who need more structure and guidance. Ultimately, the findings underscore the promise of token-based systems, while also highlighting the need for

thoughtful design and complementary support. More targeted research is needed to fully understand how flexible policies interact with other course structures (such as the requirement to attempt all programs in Fall 2019) and student characteristics, but this analysis shows that token based system provides a sizable improvement over the punitive policy and it is effective for all learners overall seen by the rise in both final exam grades and final course grades with the token based system.

6.3 Threats to Validity

The conclusions drawn from the data analysis must be considered in light of several important limitations and confounding variables inherent in the quasi-experimental design of the study.

6.3.1 Minimum Program Submission Requirement Policy (Fall 2019)

The most significant threat to validity is the presence of a separate course policy in Fall 2019 which stated that a grade of zero on any program assignment could result in failing the course. This high-stakes policy created a powerful incentive for students to submit something for every assignment, which likely explains the very low rate of non-submissions in that semester. Its absence in Fall 2024 makes a direct comparison of non-submission rates between the two policies difficult.

6.3.2 EMRN Grading Scheme (Fall 2024)

The Fall 2024 semester also introduced a mastery-based EMRN [29] grading scheme, where students would receive a score out of 4 (Excellent, Meets Expectations, Revision Needed, Not Assessable). This shift, implemented alongside the new late policy, represented a sub-

stantial change in how student performance was assessed and is a likely confounding variable in interpreting submission behavior. Unlike traditional point-based grading systems that reward incremental improvements toward 100%, the EMRN model defined clear performance thresholds. For example, a student might receive an "Excellent" (E) after achieving a certain benchmark in functionality and correctness, even if they did not pass all reference tests. As a result, some students may have stopped iterating on their code once they had earned an "E," leading to lower final Ref Test % scores compared to semesters where full credit was only awarded for near-perfect performance and same with MR and RN. This shift in incentives, where the marginal benefit of perfecting an assignment decreased once a satisfactory threshold was met, may have contributed to deflated Ref Test % distributions in Fall 2024, particularly for programming assignments. Understanding this dynamic is important when comparing performance metrics across semesters that use different grading paradigms.

6.3.3 Differences in Cohort Size and Consent Forms

The two semesters differed significantly in their enrollment size, which, while accounted for where possible in the analysis, still represents a limitation of the study. This discrepancy largely stems from differences in the study's participation process. In Fall 2019, students were asked to provide consent only once, at the beginning of the semester. In contrast, during Fall 2024, students were asked for consent twice, once at the start of the semester and again at its conclusion. This additional step likely reduced the number of participants who completed both consent requests, thereby decreasing the overall sample size for that semester. Furthermore, because the second consent request in Fall 2024 occurred after many students had already learned their course standing prior to the final exam, it may have influenced the types of students who ultimately chose to participate, potentially introducing selection bias into the sample.

6.3.4 Pre versus Post COVID-19 Impact

In addition, the two semesters took place in markedly different educational contexts, Fall 2019 prior to the COVID-19 pandemic and Fall 2024 several years after. Shifts in instructional practices, assessment formats, and student expectations in the post-pandemic environment may have influenced both learning behaviors and participation patterns in ways that are difficult to fully control for in the analysis.

6.4 Future Work

6.4.1 More Customization Options for Instructors

In the current implementation of the EGP Broker, instructors can only define a single, uniform set of extension passes that applies across all assignments in a course. While this provides a baseline level of flexibility, it does not reflect the range of configurations instructors may want to support. Future iterations of the system could provide a richer set of customization options, including:

- **Assignment group targeting:** UI options for instructors to apply extension passes to specific subsets of assignments, rather than enforcing one configuration for an entire course.
- **Resubmission support:** Policies for resubmissions after grading or after the original deadline, which would expand the scope of how passes can be used beyond simple deadline extensions.
- **Offset-based design:** Rather than limiting the system to an "extension" versus "resubmission" binary, a more general model could use offsets (e.g., start offset, end

offset, and whether submissions must be contiguous). This approach would make it easier to express a wider variety of policies.

- **Frequency and limits:** Configurations such as "one extension per week" or "maximum two resubmissions per assignment" would allow instructors to more precisely manage student use of passes.

Each of these directions has a clear path for implementation. For example, the existing broker logic can be extended to handle assignment groups by filtering based on Canvas' assignment metadata, while offset-based designs can build on the current extension model by generalizing date calculations. Similarly, adding support for frequency limits could be integrated into the EGP Broker's tracking of extension usage without requiring significant changes to Canvas itself.

Finally, one identified gap in the current role-mapping approach is the handling of administrative users. While Canvas course-level roles such as "student" or "instructor" mapped neatly into the EGP Broker's security model, admin users fell outside this structure. Addressing this misalignment would make the tool more consistent and robust, and would ensure that system-wide roles can be cleanly integrated into the IAM (identity and access management) layer. The relative ease of leveraging Canvas' built-in security and roles suggests that these adjustments could be incorporated without fundamentally altering the EGP Broker's interoperability-first design philosophy.

6.4.2 Student-Initiated Requests and Accommodation Support

Another direction for future work is to expand the EGP Broker to support student initiated requests for additional passes. In the current design, instructors set an initial number of free passes for all students, limiting the EGP Broker to one-sided workflows. A more flexible

model would provide students with an additional tab on their dashboard, “Request Pass”, where they can submit a form to request an additional pass. This form would contain a dropdown menu populated with the pass types enabled for the course and a required text field for students to explain the reason for their request. On the instructor side, a new dashboard tab, “Pass Requests”, would present incoming student requests in a vertically scrollable view. Each request would appear as a card displays the relevant details, with options for the instructor to approve or reject the request. This workflow allows instructors to exercise discretion while maintaining a clear record of requests and outcomes within the EGP Broker. It also introduces a two-way interaction between students and instructors, supporting more nuanced and equitable accommodations in course management.

Within this broader workflow, SSD (Services for Students with Disabilities) accommodations represent a particular use case that could be integrated as an additional feature. Future versions of the EGP Broker could allow instructors to tag students as SSD on the student management page. Once tagged, these students would see a dedicated option on their “Request Pass” tab to submit an SSD-specific pass request. Unlike general free pass requests, SSD passes would be designed for time-sensitive accommodations: when approved by the instructor, the pass would be immediately and automatically applied to the relevant assignment, ensuring that the accommodation is immediately. This preserves instructor oversight while streamlining the process for students who require legally mandated accommodations.

The workflow mirrors the general pass request structure, students submit a form, instructors review and approve, but applied directly to the target assignment and possibly connected assignment platform. The underlying mechanism still relies on the existing EGP protocol for pass applications, meaning SSD requests would not require a separate technical pathway. Instead, the EGP Broker treats SSD deadlines as another policy expressed through the extension infrastructure, rather than as an exception requiring custom handling.

By unifying SSD accommodations within the same framework as general requests, the EGP Broker provides a consistent, auditable, and privacy-preserving mechanism for managing both policy-driven extensions and legally required accommodations. This design strengthens the EGP Broker's role as a comprehensive extension management tool, while also supporting compliance with institutional and legal requirements.

Bibliography

[1] 1EdTech Consortium. 1edtech security framework 1.0: Using oauth 2.0 client-credentials grant. https://www.imsglobal.org/spec/security/v1p0/#using-oauth-2-0-client-credentials-grant, May 2019. Accessed: 2025-08-07.

[2] John Aycock and Jim Uhl. Choice in the classroom. *ACM SIGCSE Bulletin*, 37(4): 84–88, 2005. doi: 10.1145/1113847.1113883.

[3] Dana Benedicto, Jordan Schwartz, Narges Norouzi, and Lisa Yan. Wip: Automated flexible extensions for improving learning equity in large scale computing classrooms. In *2024 IEEE Frontiers in Education Conference (FIE)*, pages 1–5, 2024. doi: 10.1109/FIE61694.2024.10892976.

[4] Kevin Buffardi and Stephen H. Edwards. Introducing codeworkout: an adaptive and social learning environment (abstract only). In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, page 724, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450326056. doi: 10.1145/2538862.2544317. URL https://doi.org/10.1145/2538862.2544317.

[5] Jennifer Campbell and Karen Reid. Comparing the impact of strict and flexible deadline policies. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSETS 2025, page 192–198, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400705311. doi: 10.1145/3641554.3701819. URL https://doi-org.ezproxy.lib.vt.edu/10.1145/3641554.3701819.

[6] IMS Global Learning Consortium. Learning tools interoperability (lti). https://ww

w.imsglobal.org/activity/learning-tools-interoperability, 2019. Accessed: 2025-08-07.

[7] IMS Global Learning Consortium. Lti 1.3 specification. https://www.imsglobal.org/spec/lti/v1p3, 2019. Accessed: 2025-08-07.

[8] Caio Cury. ltijs: Node.js library for lti 1.3 tool development. https://github.com/Cvmcosta/ltijs, 2020. Accessed: 2025-08-07.

[9] Docker Inc. Docker. https://www.docker.com. Containerization platform, accessed 2025-08-07.

[10] Stephen H. Edwards and Manuel A. Perez-Quinones. Web-cat: automatically grading programming assignments. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, ITiCSE '08, page 328, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580784. doi: 10.1145/1384271.1384371. URL https://doi.org/10.1145/1384271.1384371.

[11] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.d. dissertation, University of California, Irvine, 2000. URL https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[12] Gradescope. Online grading platform. https://www.gradescope.com, n.d.

[13] Dick Hardt. The oauth 2.0 authorization framework. Technical Report RFC 6749, Internet Engineering Task Force, 2012. URL https://doi.org/10.17487/RFC6749.

[14] Melissa Hills and Kim Peacock. Replacing power with flexible structure: Implementing flexible deadlines to improve student learning experiences. *Teaching and Learning Inquiry*, 10, 07 2022. doi: 10.20343/teachlearninqu.10.26.

[15] John R. Hott. Analyzing student performance with free late submission days. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 2*, SIGCSE 2024, page 1682–1683, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704246. doi: 10.1145/3626253.3635562. URL https://doi.org/10.1145/3626253.3635562.

[16] Instructure. Canvas. https://www.instructure.com/canvas, n.d. Accessed: 2025-08-07.

[17] Instructure Developer Documentation Portal. Navigation tools — canvas lms external tools (lti) placements. Online documentation. URL https://developerdocs.instructure.com/services/canvas/external-tools/lti/placements/file.navigation_tools. Retrieved August 7, 2025.

[18] Mandy Korpusik, Jordan Freitas, and John David Dionisio. Impact of late policies on submission behavior and grades in computer programming. In *2022 ASEE Annual Conference & Exposition*, 2022.

[19] Charisse Liu, Yuerou Tang, Narges Norouzi, and Lisa Yan. Wip: Examining the impact of a flexible extension policy on student learning experience in a large-scale computing course. *2024 IEEE Frontiers in Education Conference (FIE)*, pages 1–5, 2024. URL https://api.semanticscholar.org/CorpusID:276619435.

[20] Meta. React – a javascript library for building user interfaces. https://reactjs.org, n.d. Accessed: 2025-08-07.

[21] MongoDB Inc. Mongodb. https://www.mongodb.com. Database software, accessed 2025-08-07.

[22] Ngrok Technologies, Inc. ngrok. https://ngrok.com, 2024. Accessed: 2025-08-07.

[23] OpenJS Foundation. Node.js — javascript runtime built on chrome's v8 engine. https://nodejs.org. Software, accessed 2025-08-07.

[24] Clifford A. Shaffer and Stephen H. Edwards. Scheduling and student performance. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, page 331, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306973. doi: 10.1145/1999747.1999842. URL https://doi.org/10.1145/1999747.1999842.

[25] Clifford A Shaffer, Ville Karavirta, Ari Korhonen, and Thomas L Naps. Opendsa: beginning a community active-ebook project. In *Proceedings of the 11th Koli Calling International Conference on computing education research*, pages 112–117, 2011.

[26] shatkinl. An inclusive time-saver: The token tracker brings structure to flexible assignment policies. https://blogs.oregonstate.edu/inspire/2023/06/20/an-inclusive-time-saver-the-token-tracker-brings-structure-to-flexible-assignment-policies/, 2025. Accessed August 6, 2025.

[27] E. G. Snider. Late passes in canvas: One approach to structured flexibility, April 2024. URL https://blogs.bsu.edu/teaching-innovation/2024/04/10/late-passes-in-canvas-one-approach-to-structured-flexibility/. The Teaching Innovation Blog, Ball State University.

[28] StrongLoop, IBM. Express — fast, unopinionated, minimalist web framework for node.js. https://expressjs.com. Software, accessed 2025-08-07.

[29] Rodney Y. Stutzman and Kimberly H. Race. EMRF: Everyday rubric grading. https://eric.ed.gov/?id=EJ717675, January 2004.

[30] The Kubernetes Authors. Kubernetes. https://kubernetes.io. Open-source container orchestration platform, accessed 2025-08-07.

[31] Jeramey Tyler, Matthew Peveler, and Barbara Cutler. A flexible late day policy reduces stress and improves learning. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '17, page 718, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450346986. doi: 10.1145/3017680.3022439. URL https://doi.org/10.1145/3017680.3022439.

Appendices

Appendix A

IRB Approval Letter

Division of Scholarly Integrity and Research Compliance
Institutional Review Board
North End Center, Suite 4120 (MC 0497)
300 Turner Street NW
Blacksburg, Virginia 24061
540/231-3732
irb@vt.edu
http://www.research.vt.edu/sirc/hrpp

MEMORANDUM

DATE: August 14, 2025

TO: Stephen H Edwards, Bob Edmison Jr, Saketh Rajesh

FROM: Virginia Tech Institutional Review Board (FWA00000572)

PROTOCOL TITLE: Collaborative Research: Transforming Grading Practices in the CS Education Community

IRB NUMBER: **22-916**

Effective August 14, 2025, the Virginia Tech Human Research Protection Program (HRPP) determined that this protocol meets the criteria for exemption from IRB review under 45 CFR 46.104(d) category (ies) 1,2(ii),3(i)(B),4(ii).

Ongoing IRB review and approval by this organization is not required. This determination applies only to the activities described in the IRB submission and does not apply should any changes be made. If changes are made and there are questions about whether these activities impact the exempt determination, please submit an amendment to the HRPP for a determination.

This exempt determination does not apply to any collaborating institution(s). The Virginia Tech HRPP and IRB cannot provide an exemption that overrides the jurisdiction of a local IRB or other institutional mechanism for determining exemptions.

All investigators (listed above) are required to comply with the researcher requirements outlined at:

https://secure.research.vt.edu/external/irb/responsibilities.htm

(Please review responsibilities before beginning your research.)

PROTOCOL INFORMATION:

Determined As:	**Exempt, under 45 CFR 46.104(d) category(ies) 1,2(ii),3(i)(B),4(ii)**
Protocol Determination Date:	**November 10, 2023**

ASSOCIATED FUNDING:

The table on the following page indicates whether grant proposals are related to this protocol.

Date*	OSP Number	Sponsor
10/06/2022	PXGZMEJY	National Science Foundation (Title: Collaborative Research: Transforming Grading Practices in the CS Education Community)

* Date this proposal number was added.

If this protocol is to cover any other grant proposals, please contact the HRPP office (irb@vt.edu).