

StudyFlow: Smart Academic Planner

Amina Kenes, Alisher Izmukhan, Askar Aibar, Mansur Rakhimov, Shynggys Lukpan

Department of Computer Science

Nazarbayev University

Astana, Kazakhstan

amina.kenes@nu.edu.kz, alisher.izmukhan@nu.edu.kz, askar.aibar@nu.edu.kz,

mansur.rakhimov@nu.edu.kz, shynggys.lukpan@nu.edu.kz

I. EXECUTIVE SUMMARY

StudyFlow is a full-stack academic planning app that was created to alleviate the academic information fragmentation of students between various sources, such as course syllabi, registrar, email, calendars, and informal university announcement systems. The ultimate project scope of the Spring 2026 project consists of the following three deliverables: a backend service, web frontend, and a mobile app on Flutter. The project will solve a practically significant issue unique to the Nazarbayev University students: despite the abundance of general-purpose productivity tools, a single platform that meets NU-specific academic processes, grading workflow, interaction with a registrar, and the use of Google-based communication is lacking [1] [2] [3]. The three-tier architecture which was implemented over the two-semester project includes FastAPI backend, a web client written in React, a mobile client written in Flutter, and persistence written in PostgreSQL. This system combines user authentication, academic planning on a dashboard, syllabus uploading and parsing, calculating GPA and grades, event management based on calendar with conflict detection, event discovery aided by Gmail, registrar synchronisation, and review options, in a single workflow-oriented system. Along with the initially planned scope, a number of extensions were added, such as email verification, password reset feature, browsing NU events, a mobile frontend track, and an administrative module. Another significant technical change in Spring 2026 was the replacement of a hand-made NLP-based syllabus parser with the help of LLM. The original spaCy-style named entity recognition-based first parser could correctly process structured examples but failed to make consistent predictions to real university syllabi, which differed in format, phrasing, grading structure and date formatting. The ultimate design thus took on an LLM-based pipeline where the information extracted is displayed as suggestions that can be reviewed by the user instead of authoritative outputs. This modification enhanced practicability and resilience within the project boundaries and adjusted the system to the wider change towards adaptive AI-aided processing of unstructured academic texts [4]. The project was written in an iterative and Agile-inspired approach with the help of Trello to track the tasks and GitHub to collaboratively oversee the version. The

last system was implemented with the help of a cloud-based stack where Render was used as a backend, and Vercel was used as a web frontend and Neon was used as a database provider. Implementation wise, it can be characterized as a provable, integrated prototype with a large functional breadth as opposed to a production-hardened platform. The main processes, including authentication, the dashboard, uploading syllabuses, extracting using the LLM, calculating the GPA and grades, detecting conflicts in the schedule, suggesting events with the help of Gmail and submitting a review, are applied. Nevertheless, certain planned features have not been finished or implemented at all, such as full support of syllabus extraction into calendar and grade-computation processes, direct Google Calendar integration outside of ICS export, GPA-optimisation, and the validation of prerequisites. The project thus meets the key criterion of the course brief in designing, implementing and evaluating a computing based solution to an academic coordination problem that was clearly defined [5]. Simultaneously, the final report clearly recognizes the existing shortcomings of the system, such as the lack of full automation, insufficient security hardening, insufficient automated testing, and a number of integration holes between clients and backend services. These constraints are noteworthy to note, as they substantially influence the understanding of the maturity of the system and give a clear foundation to its further development. Along with these scheduled requirements, the deployed system also has a number of key extensions over and above the initial scope of Spring, such as email validation, password reset, NU events browsing, the mobile frontend track, and an administration module. These extensions are proactive and widen the scope of usefulness of the platform, but they also added complexity to the scope and partially justify why not every additional integration was done during the project duration.

II. INTRODUCTION

The academic space of university students is extremely fragmented and critical information is made available in numerous systems including the registrar systems, email communications, course syllabi and personal calendar systems. This discontinuity causes a cognitive burden, increased chances of missing deadlines, and the general efficiency of academic

TABLE I: Planned and Actual Functional Requirement Delivery

FR	Planned Requirement	Final Status
FR-1	Registration and login with email/password or Google OAuth	Implemented
FR-2	Protected dashboard with courses, grades, and events	Implemented
FR-3	PDF syllabus upload	Implemented
FR-4	LLM-based extraction of assessments, weights, and deadlines	Implemented
FR-5	User-verifiable extracted syllabus data before event creation	Implemented
FR-6	Automatic per-course grade calculators from extracted grading structure	Implemented
FR-7	Real-time grade input and projected course grade calculation	Implemented
FR-8	Dedicated GPA calculator page	Implemented
FR-9	Required future grades for user-defined GPA target	Implemented
FR-10	Calendar combining schedules, syllabus deadlines, and user-created events	Partially implemented
FR-11	Scheduling conflict detection	Implemented
FR-12	Gmail integration with LLM summary and event suggestion	Implemented
FR-13	Explicit confirmation before adding Gmail-derived events	Implemented
FR-14	Export to Google Calendar or ICS	Implemented
FR-15	Prerequisite and antirequisite checking	Deferred and not implemented
FR-16	Anonymous course and instructor reviews	Implemented

planning is reduced. In Nazarbayev University, particularly, students rely on a combination of institutional mechanisms and external mechanisms, neither of which will provide a unique workflow based solution to local academic structures. The issue that the present project seeks to achieve thus is the lack of a cohesive, unified academic planning framework that would be consistent with NU-specific procedures, such as grading policies, interaction with the registrar, and Google-based communication infrastructure [1] [2] [3]. The existing tools, despite their effectiveness as single tools, fail to promote the seamless integration process of academic data sources and the student processes. StudyFlow proposes a full stack platform, which pulls these fragmented processes together under the same platform. Academic-timetable, Gmail-based-event-discovery, syllabus parsing, GPA-tracking and registrar-synchronisation are also integrated into the system which can be accessed through web and mobile-based clients. This architecture is set up to enable it to accommodate modularity, extensibility and real world use. The following structure is taken in this report. Section 3 discusses the constraints influencing the system design. Section 4 discusses related work and solutions. Section 5 shows system architecture and approach to design. Section 6 outlines the process of implementation

and the critical decisions. Section 7 considers the system on a basis of user-centred validation. Lastly, Section 8 sums up the findings and future work.

III. IDENTIFICATION OF CONSTRAINTS

The development of StudyFlow was influenced by a range of constraints extending beyond purely technical considerations. These include economic, ethical, social, regulatory, and deployability-related factors, all of which had a direct impact on architectural and implementation decisions. From an economic perspective, the system was developed under the constraints of a student project, requiring the use of cost-effective infrastructure. As a result, the architecture prioritises open-source technologies and free-tier cloud services such as Render, Vercel, and Neon, favouring accessibility and feasibility over high-performance or enterprise-grade deployment. Ethical and privacy constraints played a central role in system design. The platform processes sensitive academic and personal data, including grades, GPA records, registrar credentials, and Gmail content. To mitigate associated risks, the system enforces restricted API scopes, explicit user consent mechanisms, and controlled data processing flows. In particular, AI-generated outputs—such as syllabus parsing results or Gmail-derived events—are presented as suggestions requiring user confirmation, ensuring transparency and user control [1] [2] [3]. Technically, one of the constraints was the fact that syllabus documents could vary significantly. Scholarly syllabi are highly heterogeneous in format, language and structure and deterministic methods of extraction are not reliable. This is the reason it was decided to substitute a custom NLP-based parser with an extraction workflow aided by an LLM [4]. The dependency on third-party services, especially Google APIs, presented regulatory and platform constraints. Authentication using OAuth, Gmail access scopes, and calendar integration needs to put pressure on system design and necessitate special attention to permissions and data flows to adhere to platform policies [1] [2] [3]. Lastly, architectural decisions were affected by deployability/sustainability constraints. The system was meant to be suitable for storage in containers and reproducible with a simplified deployment pipeline and no complicated infrastructure dependencies. Nevertheless, robustness at production level including high-level security hardening and scalability assurance is not yet within the scope.

IV. BACKGROUND AND RELATED WORK

The process of creating StudyFlow lies at the crossroads of academic planning systems, educational data mining, and natural language processing. Current solutions in this field generally deal with single-facet elements of the student experience instead of offering a workflow. Learning Management Systems (LMS) and institutional portals are more of information repositories of academic information, with no proactive planning features. Productivity applications and calendar tools are good to schedule things but have to be entered manually and they are not connected directly with academic documents and school systems. The studies in the field of course recommendation

systems show that data-driven methods can be used to facilitate academic decisions [6] [7]. As an illustration, performance modelling and curriculum analysis systems can help students to choose courses and optimise study paths [6] [7]. But these systems are usually aimed at pre-semester planning as opposed to continuous academic coordination. Information extraction tasks, such as document parsing and entity recognition have been extensively performed with the use of natural language processing. Conventional methods, including named entity recognition (NER) with systems like spaCy, are based on regular patterns and annotated datasets. These methods work well in controlled settings, but fail with very dynamic and unstructured documents like academic syllabi [4]. More recent developments in large language models such as transformer-based models like BERT have shown better performance in unstructured text processing with the help of contextual understanding [4]. This has resulted in a transition to pipelines that are assisted with LLMs, which are more accommodating to heterogeneous input formats and less domain-specific training data [4]. Unlike current systems, what StudyFlow brings to the table is not a new algorithm, but a combination of various academic features into one user-friendly workflow. The system fills a gap between the individual tools and the integrated requirements of students throughout an academic semester by integrating syllabus interpretation, scheduling, GPA tracking, email analysis, and registrar integration.

V. PROJECT APPROACH

The StudyFlow takes the form of a multi level, service based architecture that is composed of a central back-end platform and two front-end applications (web and mobile). Such design allows separation of concerns, scaling of individual components and consistency in how data is handled across user interfaces. The system has the backend service at its heart, and this service handles identity management, data persistence, business logic implementation, and external-service integration. The backend provides communication with third-party APIs, such as Google OAuth to authenticate and Gmail API to process emails and syllabus parsing services based on LLM. Long-term data storage is stored using a postgresSQL database, which stores user information, academic records, events, and system state. The user interaction layers are the frontend clients which are a React web app and a Flutter mobile app. The tasks of these clients are to make the user interface, maintain the local state, and coordinate user actions like planning a course, calculating GPA, and handling events. The clients can both talk to the backend through RESTful APIs, which guarantees a uniform service contract between platforms [8].

A. Architecture and Component Mapping

The backend is based on a modular FastAPI application, which is divided into various layers such as routers, schemas, models, CRUD operations and service modules [9]. This stratified design is conducive to maintainability and separation of request processing, access to data and business logic. Fast

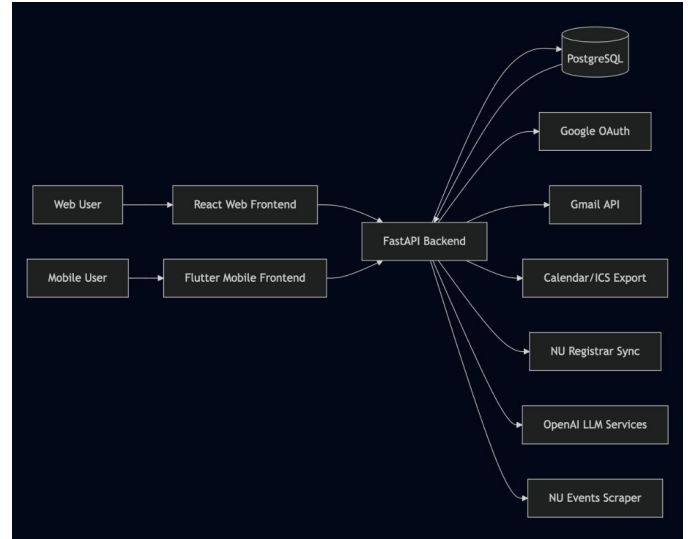


Fig. 1: Overall StudyFlow architecture showing the two client applications, shared backend services, persistent storage, and third-party integrations.

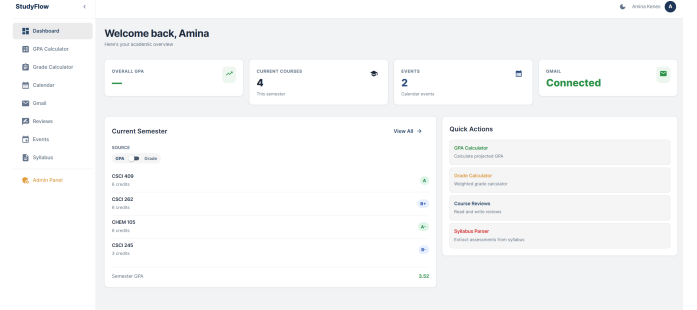


Fig. 2: Web dashboard consolidating GPA, courses, events, and integration status.

API has a lifespan context to control the application lifecycle, allowing initialisation and shutdown to be controlled. The system also undertakes a database initialisation and background service (e.g. Gmail synchronisation manager) activation during startup. Middleware is set up, such as CORS handling and session management, to enable secure communication across origins, and OAuth state management. The structure fits the best standard practices of FastAPI applications, especially dealing with shared resources and background processes that could take some time to complete [1]. React web frontend offers a complete range of features, in the form of: Authenticated dashboard access GPA and grade calculators Calendar-based planning and conflict detection Gmail integration and message processing Syllabus parsing interface Course and instructor reviews Profile management and event browsing It is designed in a way that it has a blend of global context (to perform authentication), component-level state, and lightweight persistence consisting of browser localStorage. React Router is used to implement routing and access control to ensure ability of protecting navigation flows and modular structure of pages [10] [11].

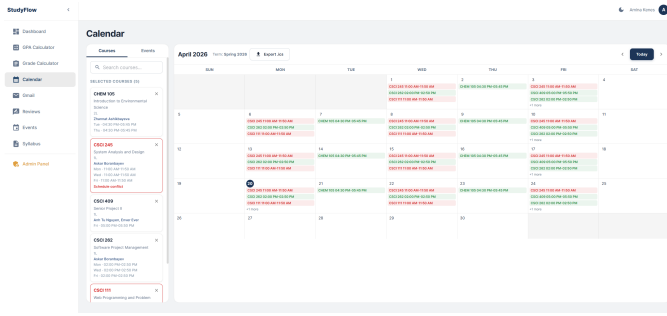


Fig. 3: Calendar planner integrating class meetings, manual events, and conflict checking.

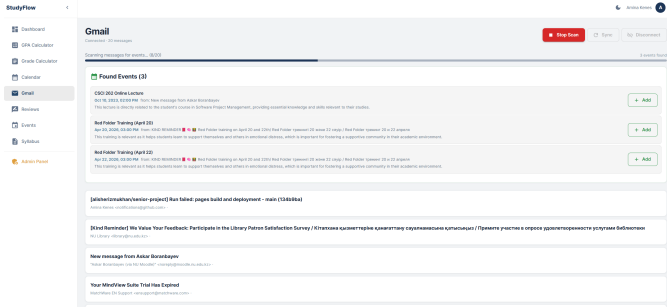


Fig. 4: Gmail-assisted event discovery with explicit user confirmation before calendar insertion.

The Flutter mobile app is a reflection of the main functionality of the web one with an adjustment to the interaction patterns of mobile devices. It uses River pod to manage state, Go router to navigate, Dio to communicate through HTTP and Shared preferences to stock local data [12] [13]. This is a technology stack of the modular and test architecture; this is based on the recommended practices of Flutter. Riverpod to a large degree has strongly typed and composable state management, which would make it a good fit to managing several independent feature modules in the application [12] [13]. On the whole, the adopted architecture represents a compromise among modularity, practicality and complexity of integration, that allows the system to cater to multi-platform applications, yet a single, backed-up backend-driven logic.

B. Syllabus and Event Workflow

The workflow of syllabus is one of the most distinctive elements of the project as it will convert the unstructured academic documents to the structured planning artefacts. The final design does not assume that the LLM output should be true [4]. Instead, back end receives plain text or uploaded PDFs, sends text extracted to the LLM parser and provides formatted recommendations, which users may take into account before creating calendar events or access downstream data [11] [12] [13]. The design is significant both in technical and ethical aspects.

This human-in-the-loop approach is two realities. To begin with, Google and calendar integrations must be explicitly trusted and have boundaries of action [8]. Second, academic

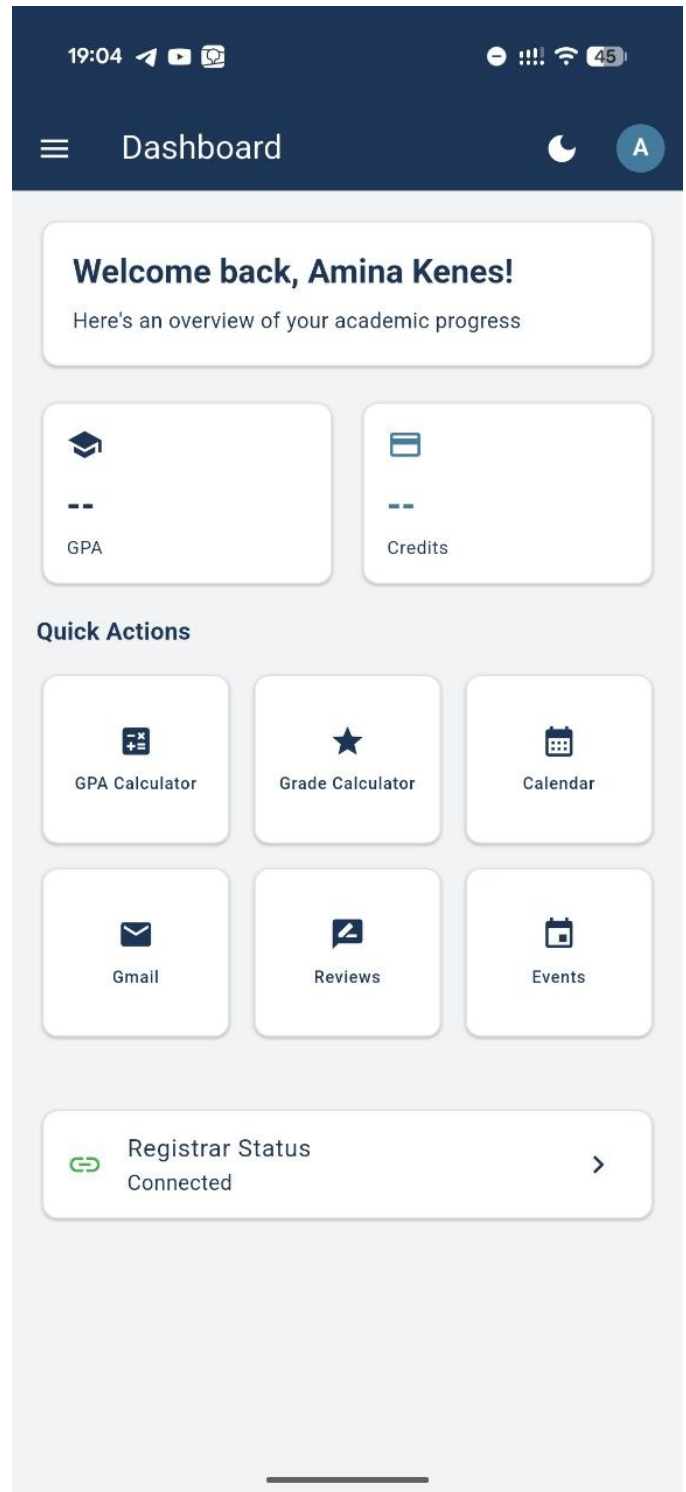


Fig. 5: Flutter mobile interface for study progress and registrar-linked academic data.

planning systems can never presume the correctness of harvesting always-right dates. The report then should present the parser as a non-binding automation engine, but not as an engaging inference mechanism [4].

TABLE II: Subsystem Responsibilities Mapping

Layer	Main Responsibilities	Key Technologies	Main Interfaces
Backend	Auth, persistence, registrar sync, Gmail sync, syllabus parsing, reviews, admin, event logic	FastAPI, SQLAlchemy, PostgreSQL, Authlib, SMTP, OpenAI API	REST endpoints (/api/v1), background Gmail sync, database models
Web frontend	Browser dashboard, scheduling, GPA/grade UI, syllabus page, Gmail page, pro-file/integration flows	React, React Router, Material UI, Axios	Calls backend REST endpoints, lightweight local state
Mobile frontend	Mobile navigation, registrar/GPA access, grade calculator, event management, Gmail and syllabus flows	Flutter, Riverpod, GoRouter, Dio, Shared-Preferences	Same backend endpoints, mobile-specific providers

VI. PROJECT EXECUTION

The introduction of the project of two semesters can be described as a continuity and redirection of strategies. The original inspiration, general product vision, and initial technological orientation, especially the idea of a tailor-made NLP-based syllabus-parser were determined in the earlier semester report. These foundations were narrowed down to a more implementation-oriented view of the world, during the Spring 2026 phase. Subsequently, the functional requirements and system scope became clear, and all the initially abstract assumptions were substituted by the decisions that were based on the empirical behaviour of the system. The biggest consequence of such refinement was the shift to a custom piping of parsing to an LLM-assisted extraction strategy [4]. The development was based on an iterative, Agile-inspired approach that included a constant review cycle inside the organization, incremental development of features and a division of tasks into modules by components of the system. Such a strategy is manifested in the maturity of the implemented system in general. Towards the end of the semester, a significant proportion of the platform was in an end-to-end usable state, on both the backend and web layers. Components such as authentication processes, protected access to dashboard, calendar based planning, Gmail message ingestion and processing, parsing syllabus interfaces, review functionality and registrar related summaries are implemented. One of the key architectural choices made was simultaneous development of the mobile application, instead of viewing it as an extension of the development. This methodology greatly enhanced the practical applicability and usability of the system as the system conforms to the pattern of interaction between users in the

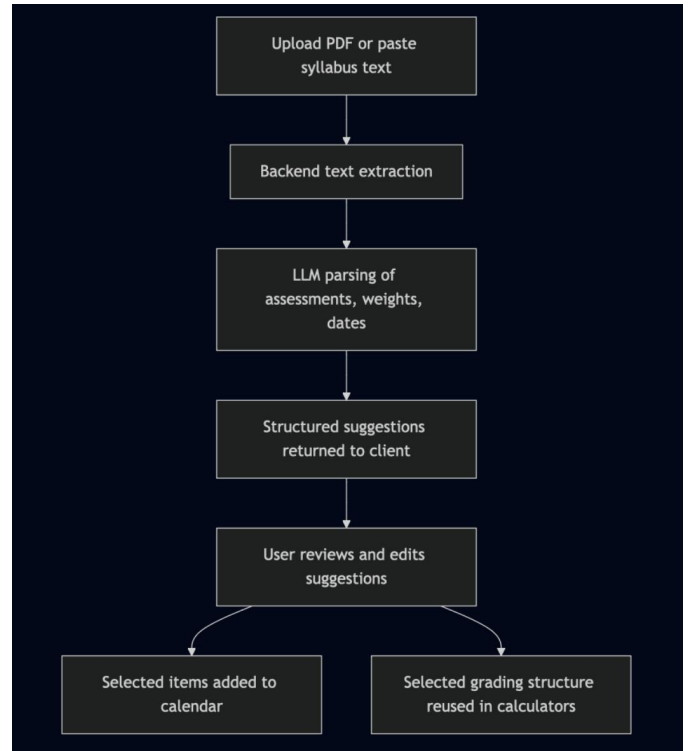


Fig. 6: User-in-the-loop syllabus processing workflow after the Spring 2026 pivot from custom NLP to LLM-based extraction.

Assessment	Date	Weight
Midterm Exam	18th	15%
Participation - Online Discussions	18th	15%
Homework Assignments	18th	15%
Team Project Experiments	18th	15%
Final Exam	18th	15%

Fig. 7: Reviewable LLM-extracted assessments and deadlines from uploaded syllabus data.

real world using various devices. But it also came with extra complexity and time constraints as it meant the need to ensure consistency over web and mobile clients as well as meet a common backend API contract. The most crucial design point in the design was during the syllabus parsing subsystem. A custom machine learning pipeline (using spaCy and standard NLP methods) was used in the first implementation. Nonetheless, the real-world testing of university syllabi empirically showed that there were significant constraints associated with the wide range of variability in the structure of documents, their formatting, and expressiveness. This transition is an engineering decision based on data, instead of being viewed as a failure. A re-evaluation of practicality showed it would not be feasible to scale up a labelled dataset and retrain the model to

obtain reliable generalisation in the project limits. As a result, the system was reconfigured as an LLM-based extraction pipeline and uses external APIs to handle unstructured and heterogeneous textual inputs to a greater extent [4]. This change of architecture had a big impact on the downstream system components. In contrast to the deterministic results that were considered in the original design, the LLM-based design yields probabilistic results, which need to be validated. This led to the redesign of workflows like calendar event generation, grade-structure propagation, to include staged processing and clear user validation. This indicates a shift in complete automation of pipelines to the human-in-the-loop interaction model, the reliability and user control are more important than complete automation –. Coworking of the subsystems is according to the modularity and technical complexity of the solution. Amina Kenes and Shynggys Lukpan were the main performers of frontend development such as user interface design and interaction flows. Askar Aibar led the backend architecture and API design, as well as integration with external services. NLP parser Mansur Rakhimov was the first to develop the custom NLP parser, and he was later to work on documentation and other supporting tasks, after the architectural pivot. The overall technical leadership which comprised of architectural decision-making, coordination of tasks and integration of subsystems was handled by Alisher Izmukhan –. The distribution is a coherent representation of individual contributions to the structural decomposition of a system. Such integration and deployment process also demonstrates the practical orientation of the project. Render, Vercel, and Neon were also used as platforms to create a cloud-based deployment pipeline, which allowed demonstrating and reproducing the system in real-time . The use of containerisation and CI-oriented practices led to the homogenous and mobile development environment. Nevertheless, even after reaching the level of a deployable and testable system, the system is not yet ready to meet the requirements of robustness on the level of production. Critical components like hardening of security, enforcement of tough server-side authentication and scalability are yet to be developed. Thus, the system can be better termed as a cloud hosted, integrated prototype as opposed to a fully production ready application.

VII. EVALUATION

The StudyFlow was evaluated according to the official course brief, which presupposes showing whether the created system is able to solve the problem outlined in the introduction, instead of indicating that the software is operating properly at the implementation level . That is why, the evaluation plan incorporated user-centred validation with development level testing to test the practical utility of the platform and the functional reliability.

A. Evaluation Methodology

An iterative user-testing process was applied to the evaluation across the lifecycle of the development as the feedback was gathered during the development lifecycle and not just

TABLE III: User Evaluation Results

Evaluation Statement	Agree	Neutral	Disagree
StudyFlow reduced the need to switch between multiple platforms	82	11	7
StudyFlow improved visibility of deadlines and academic responsibilities	78	15	7
GPA and grade calculation tools were useful for academic planning	74	17	9
I trusted AI-assisted features when manual confirmation was required	64	24	12
The overall interface was easy to understand after initial use	71	18	11
Mobile accessibility would significantly improve the usefulness of the system	76	14	10
Registrar synchronisation increased the practical value of the platform	73	19	8

at the end of the project. The study was conducted through a structured questionnaire which involved 100 participants, mainly undergraduate and masters students of the Nazarbayev University, who are the target population of the system. A set of realistic academic tasks were requested to be performed by the participants using StudyFlow. These activities involved the uploading and reviewing of syllabus information, finding future deadlines, using the GPA and the grade calculator, managing academic calendar events and looking at Gmail-based academic recommendations. These activities were chosen as they represent the main goal of the platform, i.e., the minimization of fragmented academic processes and the enhancement of the efficiency of planning in one united environment. Following interaction with the system, the participants were provided with a questionnaire that had Likert-scale and open-ended questions. The questionnaire revolved around a number of user experience dimensions, such as perceived usefulness of the platform, ease of use, clarity of interface design, platform switching reduction effectiveness, trust in AI-assisted functionalities, such as syllabus parsing and Gmail event extraction, and general satisfaction with the system.

B. Questionnaire Results

The feedback shows a rather optimistic attitude towards the main functionality of StudyFlow, especially concerning the centralisation of academic processes. A significant percentage of the respondents said that the system minimized the necessity to switch between separate systems, including the registrar, email, and personal calendars. Particularly, 82% and 78% of the respondents stated that StudyFlow decreased switching between platforms and enhanced the visibility of deadlines and academic tasks, respectively. Moreover, 74 percent of them found the GPA and grade calculator tools to be helpful in academic planning and 64 percent said that they trusted the assisted features of the system, as long as manual check was still included in the process.

Simultaneously, the assessment also indicated that the current implementation had a number of shortcomings. There was some confusion with navigation among some participants, particularly with the calendar and modules dealing with the

syllabus. Other individuals have indicated that mobile accessibility played a crucial role in day-to-day academic activities and that desktops-only workflow would not be long-term viable. Some remarks were also made that the integration with the registrar system was not as good yet that it could be considered as fully automatic or seamless. Improvements are to be made based on evaluation. Some critical design and implementation decisions were made using the results of the evaluation. Feedback about complexity of navigation resulted in a complete redesign of the web interface. The redesign of the dashboard consisted of changing the dashboard layout, enhancing the boundaries between feature modules, reinforcing the visual hierarchy of information, and streamlining interaction flows on the calendar and syllabus pages. Consequently, subsequent internal testing revealed usability and enhanced clarity of navigation. Mobile accessibility was also noted to be important in the evaluation. According to the responses of many participants, academic planning activities are often done on smartphones, especially when they check deadlines, check schedules, or reply to the suggestions displayed in the email. To address this requirement, a mobile application based on Flutter was created simultaneously with the web platform enabling the system to enable more flexible and on-the-go interaction. The other significant enhancement that was affected by the user feedback was the increased registrar synchronisation capability. Respondents were in agreement on the need to be more closely integrated to official academic records. This contributed to better synchronisation of course schedules and academic records so the platform can add more authoritative university data to GPA-related views and the general academic planning. Lastly, the assessment strengthened the need to have controlled automation. The participants were mostly open to AI-assisted functionality, although those systems that would retain user control were favored. This affirmed the worth of design choices like needing confirmation prior to the addition of Gmail-derived events, permitting users to examine extracted syllabus information before it is utilized, and displaying editable results rather than non-editable, automatically committed ones.

C. Development Check and User-Centered Test

Besides user-centred testing, there were development level testing conducted during the implementation process. These encompassed manual quality assurance, unit testing of the backend components, integration testing on the frontend and backend and regression testing after the addition of major features. These tests need to be done in order to ensure the stability and functional accuracy of software in the development process. Nevertheless, the difference between development testing and user-centred evaluation should be made. Development testing considers whether the software is acting as it should to proceed with implementation. User-centred evaluation on the other hand is concerned with the issue of whether the system is the solution to the problem of academic planning to the intended users. They are both required, but have different purposes. The latter is the only

one that promotes the argument that StudyFlow can be used to solve the issue that was outlined in the introduction.

D. Evaluation Limitations

Though the assessment had promising results, a number of weaknesses must be noted. The evidence at hand lacks a demographic breakdown of the participants, standardised measures of usability like System Usability Scale scores, quantitative performance data including latency or extraction accuracy and statistical significance analysis. Consequently, although the results are highly indicative of the perception of the system as useful and effective in centralising academic information, the strength of any claim on the quantitative efficacy is weak.

E. Summary of Findings

Comprehensively, the analysis shows that StudyFlow enhances the transparency and control of academic resources by minimizing fragmentation in different systems and displaying the essential planning capabilities in one integrated system. The awareness of deadlines and the clarity of planning was always improved by the users, and the evaluation process resulted in the significant enhancement of interface design, mobile access, integration with the registrar, and the design of workflows with the use of AI. These results confirm the hypothesis that the system successfully copes with the issue stated in the introduction, as well as discovering new spheres that need to be developed.

VIII. CONCLUSION AND FUTURE WORK

A. Conclusion

StudyFlow illustrates that an academic planning system at the university level is viable and of practical worth to a university, by integrating various disparate student processes into a cohesive system. Instead of providing a specific single-purpose technical feature, the main contribution of the project is a combination of heterogeneous capabilities, such as syllabus interpretation, academic scheduling, registrar-linked progress tracking, event discovery based on Gmail, and peer review systems, in a coherent and user-focused system that meets the requirements of Nazarbayev University students. In that regard, the project meets the main goal of the course brief to design, implement and evaluate a computing-based solution to a clearly established academic coordination problem. The architectural breadth of the system, including interoperable backend, web and mobile layers, based on the shared service infrastructure, is also one of its main strengths. Such a multi-platform integration has a tremendous impact on accessibility and is indicative of actual usage trends, in which students use academic tools on multiple platforms. Furthermore, the project also has a high level of congruence between the system design and user requirements most notably in its centralisation, ease of use and managed automation. The shift in syntactic method to an NLP-based syllabus parser to an LLM-aided extraction mechanism was a critical design milestone. This juncture is indicative of an evidence-based, mature engineering choice,

as opposed to a constraint. Empirical experimentation found that the unstructured and variability of actual academic syllabi could not be reliably operated upon with traditional machine learning approaches within the project constraints. With the implementation of an LLM-based pipeline, it became possible to process more robustly and flexibly, and the implementation of user verification guaranteed reliability and transparency of system outputs. Simultaneously, it is necessary to note that the system is a showcased and combined prototype and not a solution that can be implemented. Some of the features that were intended to be implemented are not fully implemented yet and some areas of the architecture, especially in security, full automation and system robustness need to be refined. The identification of these limitations is important in the proper positioning of the effort of the project in both an academic and practical context.

B. Future Work

The development directions in the future are shown by the existing system limitations and the identified gaps in the functionality, as opposed to speculative feature extensions. To start with, the site would require the integration of the current subsystems to have a complete closed-loop academic planning process. The products of the syllabus parsing, in particular, must be more fluidly feed forward into calendar event structures and grade-calculator structures. Besides, the feature of GPA-target optimisation that has not been implemented yet should be either brought to life with a clear methodology or be out of the scope of user expectations to ensure that the capabilities of the system and the interface design are kept at par. Second, it needs to be enhanced with calendar interoperability. Although the existing system accommodates the ICS export, the next-generation should be a direct integration with Google Calendar with proper API scopes and safe consent flows [1]. This would make it more usable as it would allow real-time synchronization instead of export-import processes, which are manual. Third, it is possible to introduce the system with prerequisite and antirequisite checking as the logical development of the platform. This functionality would help in the larger objective of helping in long term academic planning, enabling students to make informed choices in terms of course selection and progression. Fourth, the platform needs additional hardening to production level. This involves the introduction of uniform authorization on the server side, less dependence on identity headers provided by the clients, improved consistency in API contracts, observability and logging, and increasing the test coverage in backends and frontends, among others –. These enhancements are key to the process of upgrading a prototype system to a dependable and scalable application. Fifth, it needs to continue developing the mobile application to reach the same level as the web platform. Among the priority areas, Google OAuth integration, eliminating device-specific assumptions, and a full-fledged testing suite to guarantee stability and usability across devices are important [12] [13] [3]. Lastly, the method used to evaluate needs to be reinforced in the future work. A

stricter assessment would involve a well-defined questionnaire measure, a carefully analyzed demographic, as well as a presentation of such quantitative measures as the usability rates and task performance indicators. It would also be possible to make comparative studies with the existing student workflows, which would enable a more robust claim of the impact of the system on the academic organisation and deadline awareness.

REFERENCES

- [1] Google, “Google calendar api overview and choose google calendar api scopes.” Accessed: Apr. 20, 2026.
- [2] Google, “Gmail api overview.” Accessed: Apr. 20, 2026.
- [3] Google, “Using oauth 2.0 to access google apis and using oauth 2.0 for web server applications.” Accessed: Apr. 20, 2026.
- [4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.
- [5] W3C, “Web content accessibility guidelines (wcag) 2.2,” Oct. 2023. Accessed: Apr. 20, 2026.
- [6] S. Algarni and F. Sheldon, “Systematic review of recommendation systems for course selection,” *Machine Learning and Knowledge Extraction*, vol. 5, no. 2, pp. 560–596, 2023.
- [7] S. Atalla, M. Daradkeh, A. Gawanmeh, H. Khalil, W. Mansoor, S. Miniaoui, and Y. Himeur, “An intelligent recommendation system for automating academic advising based on curriculum analysis and performance modeling,” *Mathematics*, vol. 11, no. 5, p. 1098, 2023.
- [8] B. Desruisseaux, “Internet calendaring and scheduling core object specification (icalendar),” tech. rep., 2009.
- [9] FastAPI, “First steps and lifespan events.” Accessed: Apr. 20, 2026.
- [10] React, “Managing state.” Accessed: Apr. 20, 2026.
- [11] React Router, “Routing and route.” Accessed: Apr. 20, 2026.
- [12] Flutter, “Flutter documentation.” Accessed: Apr. 20, 2026.
- [13] Riverpod, “Getting started.” Accessed: Apr. 20, 2026.