**Software Requirements Specification**

**for**

Align

**Version 1.0 approved**

**Prepared by: Liv Anderson**

**Zo Carroll**

**Siaka Diarra**

**Meghann Manson**

**Kursten Massey**

**February 24th, 2025**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# **Introduction**

## **Purpose**

*<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.>*

## **Document Conventions**

*<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>*

## **Intended Audience and Reading Suggestions**

*<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>*

## **Product Scope**

*<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals. Relate the software to corporate goals or business strategies. If a separate vision and scope document is available, refer to it rather than duplicating its contents here.>*

## **References**

*<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>*

# **Overall Description**

## **Product Perspective**

*<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. A simple diagram that shows the major components of the overall system, subsystem interconnections, and external interfaces can be helpful.>*

## **Product Functions**

*<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary (such as a bullet list) is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, is often effective.>*

## **User Classes and Characteristics**

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>*

## **Operating Environment**

*<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.>*

## **Design and Implementation Constraints**

*<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).>*

## **User Documentation**

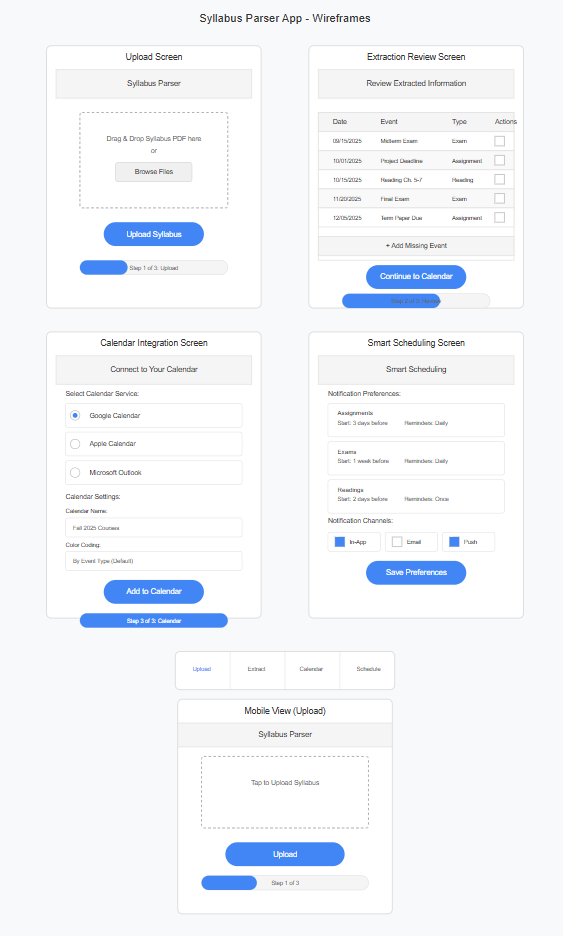
*<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>*

## **Assumptions and Dependencies**

*<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>*

# **External Interface Requirements**

## **User Interfaces**

**

## **Hardware Interfaces**

*<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>*

## **Software Interfaces**

*<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>*

## **Communications Interfaces**

*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

# **System Features**

## Syllabus Date Extraction

4.1.1 Description and Priority

The system shall accurately extract important dates, deadlines and associated contextual information from syllabi. This feature is **High priority** with the following ratings:

* Benefit: 9 (direct core functionality)
* Penalty: 9 (application cannot function without this capability)

4.1.2 Stimulus/Response Sequences

1. User uploads a syllabus
   1. System validates the format
   2. System processes the document for text extraction
2. System identifies dates and associated events
   1. System parses text for date formats and deadline language
   2. System categorizes events (assignments, exams, projects)
   3. System extracts contextual information (course codes, descriptions)
3. System presents extracted data to user for verification
   1. User confirms extraction accuracy
   2. User makes correction if needed

4.1.3 Functional Requirements

REQ-1: The system shall support uploading PDF documents up to 50MB in size

REQ-2: The system shall extract dates and events from uploaded syllabus document

REQ-3: The system shall identify and categorize at least five types of academic events (exams, assignments, readings, lectures)

REQ-4: The system shall recognize multiple date formats, including MM/DD/YYYY, Month DD, YYYY, and relative dates (e.g., “due next Monday”)

REQ-5: The system shall extract associated contextual information including course code, event description, and importance indicators

REQ-6: The system shall provide a verification interface allowing users to review and manually correct extracted information

## Calendar Integration

4.2.1 Description and Priority

The system shall create and integrate calendar events from extracted syllabus data into users’ preferred calendar platforms. This feature is **High priority** with the following ratings:

* Benefit: 8 (direct core functionality)
* Penalty: 8 (application would lose value without this)

4.2.2 Stimulus/Response Sequences

1. System generates calendar entries from extracted data
   1. System formats entries with appropriate details and context
   2. System prepares batch of events for calendar addition
2. User reviews generated calendar events
   1. User approves or modifies entries
   2. User selects target calendar
3. System adds approved events to user’s calendar
   1. System authenticates with calendar service
   2. System creates events via API
   3. System confirms successful addition

4.2.3 Functional Requirements

REQ-7: The system shall generate calendar events with title, date, time, and description fields populated from extracted syllabus data

REQ-8: The system shall integrate with at least three major calendar platforms (Google Calendar, Apple Calendar, Microsoft Outlook)

REQ-9: The system shall maintain all academic context from the syllabus in the created calendar events

REQ-10: The system support bulk addition of events to calendars with appropriate duplicate detection

REQ-11: The system shall maintain synchronization between the app and connected calendar services

## Smart Scheduling

4.3.1 Description and Priority

The system shall implement intelligent scheduling features including suggested start times and adaptive notifications based on assignment types and user patterns. This feature is **Medium-High priority** with the following ratings:

* Benefit: 7 (significant value added and differentiator)
* Penalty: 6 (loss of competitive advantage if not implemented)

4.3.2 Stimulus/Response Sequences

1. System analyzes extracted assignment details
   1. System categorizes assignment type and complexity
   2. System evaluate current calendar load
   3. System calculates suggested start dates
2. User reviews suggested start times
   1. User accepts or adjusts recommendations
   2. User configures notification preferences
3. System implements notification schedule
   1. System sets up graduated reminders
   2. System monitors user interaction with notifications
   3. System adapts future notification timing based on feedback

4.3.3 Functional Requirements

REQ-12: The system shall calculate suggested start times for assignments based on event type, estimated completion time, and existing calendar commitments

REQ-13: The system shall maintain all academic context from the syllabus in the created calendar events

REQ-14: The system shall implement an AI-driven notification system that adapts timing based on assignment type and user behavior.

REQ-15: The system shall allow users to customize notification preferences by course, event type, and time windows

REQ-16: The system shall consider at least five factors when suggesting start times (assignment type, point value, existing workload, course difficulty, and historical patterns).

REQ-17: The system shall provide multiple notification channels (in-app, email, push notification).

REQ-18: The system shall implement a learning algorithm that improves start time suggestions based on user feedback and behavior.

# **Other Nonfunctional Requirements**

## **Performance Requirements**

1. The system shall process a standard syllabus document (up to 10MB) within 30 seconds from upload completion to data extraction.
2. The calendar integration feature shall complete the addition of events to user calendars within 15 seconds of user confirmation.
3. The system shall maintain a maximum response time of 2 seconds for all user interface interactions under normal operating conditions.
4. The system shall process a standard syllabus document in under 30 seconds

## **Safety Requirements**

1. The system shall implement warning mechanisms to prevent accidental deletion of calendar events.
2. The system shall prevent creation of excessive calendar notifications that could overwhelm users by implementing reasonable limits based on event frequency and importance.
3. The system shall maintain audit logs of all syllabus processing activities to ensure traceability in case of data discrepancies or errors.

## **Security Requirements**

1. The system shall authenticate users through industry-standard OAuth 2.0 protocols when connecting to third-party calendar services.
2. The system shall encrypt all stored syllabus data and extracted information using AES-256 encryption.
3. The system shall automatically delete uploaded syllabi from storage within 24 hours after processing unless explicitly saved by the user.

## **Software Quality Attributes**

1. The system shall achieve 99.5% uptime during academic semesters (excluding scheduled maintenance).
2. The system shall be compatible with all major web browsers (Chrome, Safari, Firefox, Edge).

## Business Rules

1. Only users who uploaded a syllabus shall have access to view, modify, or delete the extracted data and associated calendar events

# **Other Requirements**

*<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>*

**Appendix A: Glossary**

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: To Be Determined List**

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*