Software Requirements Specification

for

<Project>

Version 1.0 approved

Prepared by <author>

<organization>

<date created>

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

The purpose of this Software Requirements Specification (SRS) document is to outline the requirements and specifications for the [Product Name], which is the result of our final year project. This SRS document pertains to Version 1.0. It comprehensively defines the scope of the [Product Name], encompassing all features and functionalities developed as part of our project. The SRS details the requirements for the entire system, including its constituent subsystems, ensuring a clear understanding of the project's objectives and functionality.

The [Product Name] aims to address the diverse needs of university students and faculty by providing a unified and intelligent platform for academic and financial management. This SRS serves as a critical reference point for all stakeholders involved in the development, ensuring that the final product aligns with the specified requirements and user expectations.

## 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

This SRS document is intended to be a comprehensive reference for a diverse range of stakeholders involved in the [Product Name] project. The document caters to the following types of readers:

* **Developers:** Developers (Members of the FYP Project Team) will find detailed information about the project's functional requirements, system architecture, and design constraints in this document.
* **Supervisor:** Supervisor will benefit from understanding the project's scope, constraints, and dependencies as well as the schedule outlined in the Gantt chart.
* **Testers:** Testers will use this document to ascertain the functional requirements and performance criteria necessary for testing the [Product Name].
* **Documentation Writer:** Writer will find information about user documentation requirements in this document, helping them create user guides and manuals.
* **Marketing Staff:** Marketing staff of Sukkur IBA can gain insights into the product's features and user classes, which can aid in crafting marketing materials.
* **Users:** While not typically direct readers of the SRS, users may refer to this document to understand the system's capabilities and constraints.

**Document Organization:**

The SRS is organized into sections, with each section addressing a specific aspect of the project.

It begins with an Introduction, providing an overview of the document's purpose and the product it covers.

The subsequent sections follow a logical sequence, covering aspects such as the project's scope, functional and nonfunctional requirements, external interfaces, and other essential details.

Readers are encouraged to start with the overview sections, which provide a high-level understanding of the project, and then delve into sections most pertinent to their roles and interests.

The sequence for reading the document, starting with the overview sections and proceeding to sections most pertinent to each reader type, is as follows:

* **Introduction:** For a broad understanding of the project's purpose and scope.
* **Overall Description:** Useful for project managers, developers, and testers to grasp the project's context and constraints.
* **External Interface Requirements:** Relevant to developers and testers dealing with user interfaces, hardware, and software interactions.
* **System Features:** Valuable for developers and testers to understand functional requirements.
* **Other Nonfunctional Requirements:** Pertinent to developers, testers, and project managers, providing insights into performance, security, and quality attributes.
* **Other Requirements:** For additional details that may not fit into the previous sections.
* **Appendices:** Resources such as a glossary and analysis models.

<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

The [Product Name] is a comprehensive software solution designed to cater to the diverse needs of university students and faculty members. Its purpose is to provide an integrated platform that combines academic and financial management with the assistance of a virtual AI-driven companion. The key objectives and goals of the [Product Name] project are as follows:

* **Academic Management:** The [Product Name] aims to simplify academic life by offering features such as timetable extraction from university emails, centralized access to course information, assignment deadlines, and exam schedules. It intends to streamline academic planning and organization.
* **Financial Management:** This software aspires to help students effectively manage their finances. It will provide tools for budget tracking, expense management, and financial goal setting. The objective is to promote financial responsibility among the student community.
* **AI-Powered Assistance:** The virtual AI assistant embedded in the [Product Name] is designed to enhance the user experience. It will offer personalized academic guidance, financial insights, reminders, and general assistance. The goal is to create an intelligent companion that provides valuable support.
* **Efficiency and Convenience:** The [Product Name] project seeks to offer convenience and efficiency in academic and financial management, reducing the administrative burden on students and faculty. It is aligned with the goal of simplifying complex processes and providing users with more time for their academic pursuits.

**Benefits:**

The implementation of the [Product Name] is expected to bring forth several benefits:

* Enhanced User Experience: Students and faculty will have a user-friendly, consolidated platform to manage their academic and financial affairs.
* Improved Academic Performance: Access to timely academic information and the support of an AI assistant can enhance academic performance and reduce stress.
* Financial Responsibility: Students will develop financial planning skills and manage their budgets more effectively, preparing them for future financial challenges.
* Operational Efficiency: The university can benefit from streamlined administrative processes and enhanced communication with students.

While this project is within the academic context of a university, it can be linked to broader educational institution goals. These might include improving the overall student experience, promoting academic success, and preparing students for their future careers. The [Product Name] aligns with these goals by providing a solution that enhances both the academic and financial aspects of students' lives, ultimately contributing to their success and well-being.

For a more detailed description of the project's scope and objectives, please refer to the [Product Name] Project Proposal document.

<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

[Product Name] Project Proposal Document

Author: FYP 20F-33 Team

Version: 1.0

Date: 20 September, 2023

Source: <https://github.com/20F-33-FYP-Team-Sukkur-IBA-University/Proposal>

Description: The Project Proposal Document provides a detailed overview of the project's vision, background, objectives, and scope. It offers a more comprehensive perspective on the [Product Name] project.

<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

The [Product Name] stands as a self-contained and innovative software solution developed as part of our final year project. It is not a member of a product family, nor is it intended as a direct replacement for any existing systems. Instead, it serves as a unique and standalone product designed to address specific needs within the university environment.

**Product Context:**

The [Product Name] exists within the context of university life, targeting both students and faculty. It aims to enhance the academic and financial management experiences of users by providing a consolidated platform for these purposes.

External Interfaces:

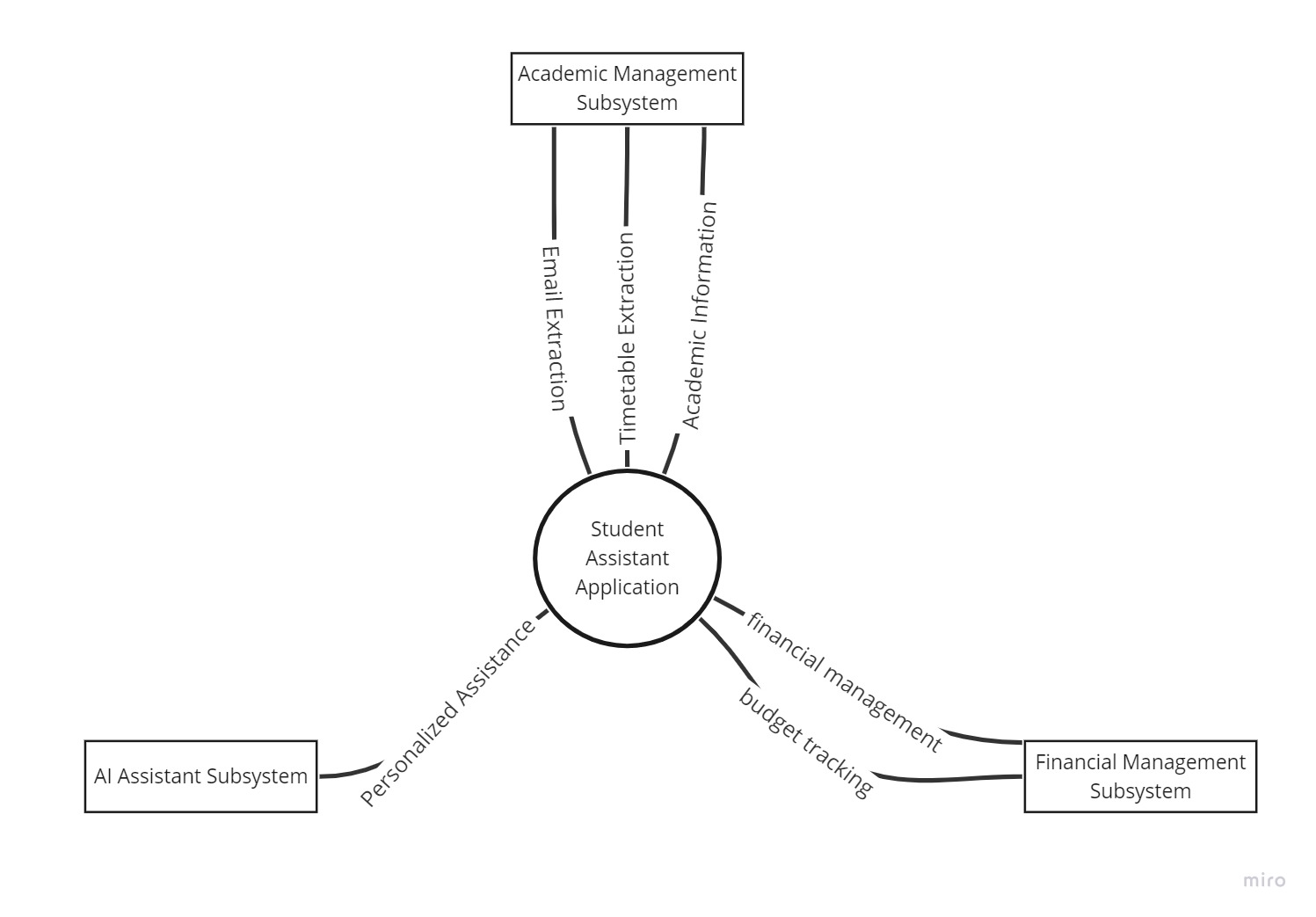
The [Product Name] interfaces with external systems in the following ways:

* **User Interfaces:** The software provides user interfaces for students and faculty to interact with the system. These interfaces enable users to access academic schedules, track budgets, and interact with the virtual AI assistant.
* **Hardware Interfaces:** While the [Product Name] primarily operates as a software solution, it may interact with standard hardware components such as personal computers, smartphones, and tablets.
* **Software Interfaces:** The system may have software dependencies, particularly when it comes to AI components or external data sources, which it communicates with as needed.

**Subsystem Interconnections:**

The [Product Name] comprises several interrelated subsystems that work cohesively to deliver its functionalities:

* **Academic Management Subsystem:** This subsystem includes features like timetable extraction, academic information management, and email extraction using AI.
* **Financial Management Subsystem:** This component allows for budget tracking and financial management.
* **AI Assistant Subsystem:** The virtual AI assistant is a central part of the [Product Name], offering personalized assistance to users.



<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

* **Academic Management Functions:**
  + Timetable extraction from university emails.
  + Email Summaries
  + Centralized access to course information.
  + Deadlines and exam schedules management.
* **Financial Management Functions:**
  + Budget tracking and expense management.
  + Financial goal setting and tracking.
  + Expense categorization and analysis.
  + Generate financial reports and summaries.
  + Data visualization for easy comprehension.
* **AI Assistant Functions:**
  + General user assistance.
  + Personalized academic and financial insights.
  + Task reminders and notifications.
  + Integration with academic and financial features.
* **User Profile Management:**
  + User registration and login.
  + Profile customization and personalization.
  + Security and privacy settings.
* **Communication and Notifications:**
  + Notification system for important academic dates and financial milestones.
  + Integration with university announcements and communication channels.
* **Search and Information Retrieval:**
  + Search functionality for academic resources, courses, and financial records.
  + Quick access to academic and financial information.
* **User Support and Help:**
  + Provide user support through the AI assistant.
  + Access to a knowledge base for common queries and issues.
  + Report and resolve technical problems.
* **Backup and Data Recovery:**
  + Automatic backup of academic and financial data.
  + Data recovery in case of system failures or data loss.

<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

**Students:**

* Frequency of Use: Daily or weekly.
* Product Functions: Use academic management and financial management features extensively.
* Technical Expertise: Varies from basic to advanced.
* Educational Level: Typically undergraduates or postgraduates.
* Experience: May be experienced with academic tools and personal finance management.

**Administrators:**

* Frequency of Use: Periodic for system maintenance and support.
* Product Functions: Ensure system integrity, troubleshoot issues.
* Technical Expertise: Advanced, with knowledge of system administration.
* Educational Level: Varied but often with technical backgrounds.
* Experience: Experienced in system administration.

**Support and Helpdesk:**

* Frequency of Use: As needed for user support.
* Product Functions: Assist users in troubleshooting and resolving issues.
* Technical Expertise: High, with proficiency in the [Product Name].
* Educational Level: Typically technically educated.
* Experience: Experienced in user support.

**Developers and Technical Team:**

* Frequency of Use: During development and maintenance.
* Product Functions: Work on system maintenance and updates.
* Technical Expertise: Advanced, with knowledge of software development.
* Educational Level: Technical backgrounds in computer science or related fields.
* Experience: Experienced in software development.

While all user classes are important, student class is the primary user class for this product. They are the core users who will rely on the academic and financial management features. Other user classes, like administrators, support, and developers, play vital roles in system maintenance and user support.

<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

* **Hardware Platform:**
  + Android-based smartphones and tablets with a minimum of 2GB of RAM and dual-core processors with optimal speed.
* **Operating System and Versions:**
  + Android 8.0 (Oreo) or later.
* **Development Tools:**
  + Android Studio for Android app development using Kotlin.
  + Python programming language for data analysis and AI components.
  + Java Runtime Environment (JRE) for the AI component.
* **Data Analysis and AI Integration:**
  + Python libraries for data analysis and machine learning, such as NumPy, Pandas, TensorFlow, and scikit-learn.
  + Integration with Python-based AI models and components.
* **Email Extraction:**
  + Integration with the Gmail API for email extraction from Gmail accounts, requiring OAuth2 authentication.
  + Use of ChatGPT AI for summarisation of emails.
* **Internet Connectivity:**
  + Reliable and secure internet connectivity is essential for real-time features and updates, as well as for interfacing with external services.
* **Screen Resolutions:**
  + The application is designed to support a range of screen resolutions to accommodate various Android devices.
* **Permissions:**
  + The application will request necessary permissions, such as access to email for the Gmail API, device storage for data management, and network access for communication.

The [Product Name] is built specifica9lly for the Android operating system, intended for use on Android-based smartphones and tablets. However, if the project is completed on optimistic estimation, the product shall also be developed for IOS based Operating Systems using XCode development Environment. It makes extensive use of Python for data analysis and AI components, enabling advanced analysis and AI-driven features within the application. The integration with the Gmail API ensures efficient email extraction from Gmail accounts.

The reliable availability of internet connectivity is essential to maintain real-time features and keep the application up to date.

<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

* **Corporate and Regulatory Policies:**

The project must adhere to the corporate policies of the university, including data security and privacy regulations. Compliance with relevant data protection and privacy laws is mandatory.

* **Hardware Limitations:**

The application's performance will be influenced by the hardware capabilities of the user's device. To ensure usability on a wide range of Android devices, the application must be optimized for devices with varying memory and processing capabilities.

* **Interfaces to Other Applications:**

The integration with the Gmail API for email extraction is subject to the availability and functionality of this external service. Any changes or limitations in the Gmail API could affect the application's email extraction capabilities.

Specific Technologies and Tools:

The use of Kotlin for Android development, Python for data analysis and AI components, and Java Runtime Environment (JRE) for the AI component are predetermined technologies.

* **Parallel Operations:**

The application should handle parallel operations efficiently, ensuring responsiveness and smooth user experience, especially when multiple features are in use simultaneously.

* **Security Considerations:**

Robust security measures must be implemented to protect user data and ensure secure communication. Encryption and secure data storage practices are mandatory.

* **Design Conventions and Programming Standards:**

The project will adhere to established design conventions and programming standards to maintain code consistency and readability. The development team will follow best practices for coding and documentation.

* **Maintenance Responsibility:**

While the development team will build the application, the customer's organization (university) will be responsible for maintaining the delivered software after deployment. This requires the development team to provide comprehensive documentation and support during the handover.

* **Compliance with Google Play Store Policies:**

If the application is intended for distribution through the Google Play Store, it must adhere to Google's policies and guidelines for app submission and updates.

These design and implementation constraints are crucial in guiding the development process and ensuring that the [Product Name] meets regulatory and organizational requirements. The development team will work within these constraints to create a secure, reliable, and efficient software solution.

<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

* **User Manuals:**

The Application is generally easy for user of any expertise to use the application and considering the user would be an undergraduate or graduate student, the user manual is **not needed**.

* **Tutorials and Help Guides:**

The tutorial would be given at the end of the project on the use of application. The AI Assistant will be given general information about the application for help guide.

* **Release Notes:**

Release notes will accompany each version of the application, documenting updates, new features, and changes. Users will be informed about improvements and enhancements.

* **Feedback and Contact Information:**

Information on how users can provide feedback, report issues, and suggest improvements will be included. Contact details for customer support and feedback channels will also be readily available.

* **Delivery Format:**

The user documentation will primarily be delivered in digital format within the application. This approach ensures easy access for users directly from their mobile devices.

* **Standards:**

The user documentation will follow standard practices for clarity, conciseness, and user-friendliness. It will be designed to provide an intuitive and user-friendly experience.

The user documentation components are tailored to help users maximize their experience with the [Product Name]. They will be easily accessible within the application, offering clear and concise guidance to enhance user proficiency.

<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

* **Assumptions:**
* **Third-Party Components:** It is assumed that third-party components and libraries used in the development process will be compatible with the application. Any unforeseen incompatibilities or changes in third-party components could affect the project.
* **Stable Gmail API:** We assume that the Gmail API will remain accessible and stable throughout the development and operational phases of the application. Any changes or unavailability of the Gmail API may impact the email extraction feature.
* **Consistency in Operating Environment:** We assume that the Android operating environment, including hardware and software, will remain consistent and continue to support the application. Changes in the Android ecosystem that affect compatibility may require adjustments.
* **User Data Privacy Compliance**: It is assumed that the university's data protection and privacy policies, as well as relevant laws and regulations, will not undergo significant changes that would necessitate a reevaluation of the application's data handling and security measures.
* **University’s Email Format:** It is assumed that the email formats for the courses and timetable will not undergo significant changes that destruct the assumed development.
* **Dependencies:**
* **Gmail API Integration:** The project is dependent on the successful integration of the Gmail API for email extraction. Any disruptions or changes to the Gmail API may impact the email extraction functionality.
* **Third-Party Libraries:** The use of third-party libraries, particularly for Python-based data analysis and AI components, creates a dependency on the availability and compatibility of these libraries.
* **Data Connectivity:** The application depends on a reliable internet connection to provide real-time features and updates. Any disruptions in internet connectivity could affect the user experience.
* **Android Operating System:** The project is dependent on the Android operating system and its versions remaining compatible with the application. Changes to Android that affect compatibility must be addressed.
* **University Policies and Regulations:** The project's compliance with university data protection, privacy policies, and relevant laws is a critical dependency. Any changes in these policies may necessitate adjustments to the application.

**Note:** Some dependencies, such as Gmail API integration and third-party libraries, are also assumptions to a certain extent, as they depend on external factors that the project team cannot control.

<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

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## 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

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

4.1.1 Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

4.1.2 Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

4.1.3 Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

REQ-1:

REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# 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.>