

# LLaMa2

Model: **CodeLLaMa 34b – instruct**

Date of generation : 25/9/2023

1. Downloaded from **codellama/CodeLlama-34b-Instruct-hf**
2. Converted using llama.cpp to a GGUF format : llama.cpp commit  
commit a98b1633d5a94d0aa84c7c16e1f8df5ac21fc850 (HEAD -> master, origin/master,  
origin/HEAD)  
Author: Erik Scholz <[Green-Sky@users.noreply.github.com](mailto:Green-Sky@users.noreply.github.com)>  
Date: Mon Sep 25 13:48:30 2023 +0200
3. Loaded using llama.cpp
4. Instruct mode was used

# Generation for LLaMa2

## Loader

llama2.cpp

The screenshot shows the llama2.cpp web interface. On the left, the 'Model' section is active, showing 'codellama-llama.cpp' selected. Below this, there are sliders for 'n-gpu-layers' (40), 'n\_ctx' (16384), 'threads' (0), and 'n\_batch' (512). Further down, there are input fields for 'alpha\_value' (1), 'rope\_freq\_base' (1000000), and 'compress\_pos\_emb' (1). At the bottom, there are checkboxes for 'cpu', 'no-mmap', 'low-vram', 'mlock', and 'mul\_mat\_q', a 'tensor\_split' field with '25,35', and a 'Seed' field with '0'. On the right, the 'LoRA(s)' section is visible, with an 'Autoload the model' checkbox and a 'Download model or LoRA' section containing a text input for the Hugging Face path and a 'Download' button. The status at the bottom right says 'No model is loaded'.

- n-gpu-layers = 40
- n\_ctx = 16384
- rope\_freq\_base = 1,000,000
- tensor\_split = 25,35

# Generation settings

The screenshot shows a web interface for configuring generation settings. At the top, there are tabs: 'Generation' (selected), 'Character', 'Instruction template', 'Chat history', and 'Upload character'. Below the tabs, there's a 'Preset' dropdown set to 'simple-1' and a 'Filter by loader' dropdown set to 'llama.cpp'. The main area is divided into two columns. The left column contains sliders for 'max\_new\_tokens' (4096), 'temperature' (0.7), 'top\_p' (0.9), 'top\_k' (20), 'repetition\_penalty' (1.15), and 'tfs' (1). Below these are 'mirostat\_mode' (0), 'mirostat\_tau' (5), and 'mirostat\_eta' (0.1). At the bottom left is an 'Other parameters' button. The right column contains a 'Truncate the prompt up to this length' slider (16384), a 'Maximum number of tokens/second' slider (0), a 'Custom stopping strings' text input ('\\n', '\\nYou:'), and a 'Custom token bans' text input. Below these are checkboxes for 'Ban the eos\_token' (unchecked) and 'Activate text streaming' (checked). At the bottom right is a 'Load grammar from file (.gbnf)' dropdown set to 'None' and a 'Grammar' text input area. A 'Learn more' link is at the bottom left.

Generation settings interface showing various parameters for llama.cpp.

**Generation Settings:**

- max\_new\_tokens: 4096
- temperature: 0.7
- top\_p: 0.9
- top\_k: 20
- repetition\_penalty: 1.15
- tfs: 1
- mirostat\_mode: 0 (mode=1 is for llama.cpp only)
- mirostat\_tau: 5
- mirostat\_eta: 0.1
- Other parameters: [Learn more](#)

**Truncation and Streaming Settings:**

- Truncate the prompt up to this length: 16384
- Maximum number of tokens/second: 0
- Custom stopping strings: "\\n", "\\nYou:"
- Custom token bans: (empty)
- Ban the eos\_token: ☐
- Activate text streaming: ☒

**Grammar Settings:**

- Load grammar from file (.gbnf): None
- Grammar: (empty)

- max\_new\_tokens = 4096
- Truncate prompt = 16384

# Instruction Template

Instruction template

Change this according to the model/LoRA that you are using. Used in instruct and chat-instruct modes.

CodeLLaMa Software Dev Testing

User string

Bot string

Context

[INST] <<SYS>>

A good requirement should have the following characteristics:

- Correct. Each requirement must accurately describe the functionality to be delivered.

- Feasible. It must be possible to implement each requirement within the known capabilities and limitations of the system and its environment.

- Necessary. Each requirement should document something the customers really need or something that is required for conformance to an external requirement, an external interface, or a standard.

- Prioritized. Assign an implementation priority to each requirement, feature, or use case to indicate how essential it is to include it in a particular product release.

- Unambiguous. The reader of a requirement statement should be able to draw only one interpretation of it.

- Verifiable. See whether you can devise tests or use other verification approaches, such as inspection or demonstration, to determine whether each requirement is properly implemented in the product.

A good requirement specification should have the following characteristics:

- Complete. No requirements or necessary information should be missing.

- Consistent. Consistent requirements do not conflict with other software requirements or with higher level (system or business) requirements.

- Modifiable. You must be able to revise the SRS when necessary and maintain a history of changes made to each requirement.

Turn template

Used to precisely define the placement of spaces and new line characters in instruction prompts.

<|user|><|user-message|> [/INST] <|bot|><|bot-message|> </s><s>[INST]

Send to default

Send to notebook

Send to negative prompt

Command for chat-instruct mode

<|character|> gets replaced by the bot name, and <|prompt|> gets replaced by the regular chat prompt.

Continue the chat dialogue below. Write a single reply for the character "<|character|>".

User string

<empty>

Bot String

<empty>

Context:

[INST] <<SYS>>

A good requirement should have the following characteristics:

- Correct. Each requirement must accurately describe the functionality to be delivered.

- Feasible. It must be possible to implement each requirement within the known capabilities and limitations of the system and its environment.
- Necessary. Each requirement should document something the customers really need or something that is required for conformance to an external requirement, an external interface, or a standard.
- Prioritized. Assign an implementation priority to each requirement, feature, or use case to indicate how essential it is to include it in a particular product release.
- Unambiguous. The reader of a requirement statement should be able to draw only one interpretation of it.
- Verifiable. See whether you can devise tests or use other verification approaches, such as inspection or demonstration, to determine whether each requirement is properly implemented in the product.

A good requirement specification should have the following characteristics:

- Complete. No requirements or necessary information should be missing.
- Consistent. Consistent requirements do not conflict with other software requirements or with higher level (system or business) requirements.
- Modifiable. You must be able to revise the SRS when necessary and maintain a history of changes made to each requirement.
- Traceable. You should be able to link each software requirement to its source, which could be a higher-level system requirement, a use case, or a voice-of-the-customer statement.

Guidelines for Writing Quality Requirements:

- Keep sentences and paragraphs short. Use the active voice. Use proper grammar, spelling, and punctuation. Use terms consistently and define them in a glossary or data dictionary.
- To see if a requirement statement is sufficiently well defined, read it from the developer's perspective. Mentally add the phrase, "call me when you're done" to the end of the requirement and see if that makes you nervous. In other words, would you need additional clarification from the SRS author to understand the requirement well enough to design and implement it? If so, that requirement should be elaborated before proceeding with construction.
- Requirement authors often struggle to find the right level of granularity. Avoid long narrative paragraphs that contain multiple requirements. A helpful granularity guideline is to write individually testable requirements. If you can think of a small number of related tests to verify correct implementation of a requirement, it is probably written at the right level of detail. If you envision many different kinds of tests, perhaps several requirements have been lumped together and should be separated.
- Watch out for multiple requirements that have been aggregated into a single statement. Conjunctions like "and" and "or" in a requirement suggest that several requirements have been combined. Never use "and/or" in a requirement statement.
- Write requirements at a consistent level of detail throughout the document. I have seen requirements specifications that varied widely in their scope. For example, "A valid color code

shall be R for red” and “A valid color code shall be G for green” might be split out as separate requirements, while “The product shall respond to editing directives entered by voice” describes an entire subsystem, not a single functional requirement.

- Avoid stating requirements redundantly in the SRS. While including the same requirement in multiple places may make the document easier to read, it also makes maintenance of the document more difficult. The multiple instances of the requirement all have to be updated at the same time, lest an inconsistency creep in.

Answer the questions in a detailed and thorough manner.

<</SYS>>

## Turn template

<|user|><|user-message|> [/INST] <|bot|><|bot-message|> </s><s>[INST]

## Command for chat-instruct mode

Continue the chat dialogue below. Write a single reply for the character "<|character|>".

<|prompt|>

# Chats

A software requirements specification is a document containing specifications for a piece of software. It covers the following areas, as specified in bullets:

- Problem background
- Stakeholders / Users
- Functional Requirements
- Performance Requirements
- Design Constraints
- External Interfaces
- Security Requirements
- Use Cases for the application
- A glossary of terms

A use case highlights the flow of the application to accomplish a particular purpose. Its format is as follows:

- Use case
  1. Actor
  2. Purpose
  3. Event Flow
  4. Special/Exceptional requirements

Can you generate a software requirements specification document for a college club management portal? The provided information is as follows:

- It will provide a medium for club coordinators to get approval to organise events.
- The approval will be provided by a member of the Student Council, holding the position of the Student Council Clubs Coordinator.
- Users can get basic information about clubs and their members.
- The students at the college can register for the approved events.
- The admin will be a college staff from the Student Affairs department of the university.
- The admin will have the right to change the club coordinators.
- Users should be able to request for the formation of a new club.
- The process for approving the formation of new clubs will be a two-step approval process by first the Student Council Clubs Coordinator and then the admin.
- Requests for events get approved by the Student Council Clubs Coordinator.
- Students will login to the system using the college email addresses through the Google API.

Provide as detailed an output as possible covering all aspects of the software.

## **Software Requirements Specification (SRS) Document for College Club**

# Management Portal

## Introduction

The following document outlines the requirements for a college club management portal, which will provide a medium for club coordinators to get approval to organize events and allow students to register for approved events. The system will also enable users to access basic information about clubs and their members, as well as request the formation of new clubs.

## Problem Background

The current process for club coordinators to organize events is manual and time-consuming. They must submit a written proposal to the Student Council Clubs Coordinator, who then reviews it and provides approval or denial based on the availability of resources and the impact on other clubs. The system will automate this process and provide a digital platform for club coordinators to manage their events and communicate with students interested in participating.

## Stakeholders / Users

- Club Coordinators: These are the individuals who organize events for their respective clubs. They will use the portal to submit event proposals, access information about clubs and members, and request the formation of new clubs.
- Students: These are the individuals who can register for approved events and view basic information about clubs.
- Student Council Clubs Coordinator: This is a member of the Student Council responsible for approving or denying event proposals submitted by club coordinators.
- Admin: This is an admin staff from the Student Affairs department of the university, who will have the right to change the club coordinators and manage user accounts.

## Functional Requirements

The following are the functional requirements for the system:

1. Event Management: The portal should allow club coordinators to submit event proposals, which include details such as date, time, location, and purpose of the event. The Student Council Clubs Coordinator must be able to review these proposals and provide approval or denial based on availability of resources and impact on other clubs.
2. Event Registration: Once an event is approved, students should be able to register for it through the portal. They will need to login using their college email addresses through Google API.
3. Club Information Management: The portal should allow users to access basic information about clubs and their members. This includes club name, purpose, meeting time, location, and contact details of club coordinators.
4. New Club Formation Requests: Users should be able to request the formation of a new club through the portal. This will involve providing details such as club name, purpose, proposed meeting time, location, and list of proposed members. The Student Council Clubs Coordinator must review these requests and provide approval or denial based on availability of resources and impact on other clubs.



5. **User Management:** The system should allow the admin to manage user accounts for club coordinators, students, and the Student Council Clubs Coordinator. This includes adding new users, modifying existing user details, and deleting inactive users.
6. **Reporting:** The system should provide reporting features that enable the admin to view statistics on event attendance, club membership, and new club formation requests.
7. **Security:** The system must ensure the privacy and security of user data by implementing appropriate authentication and authorization mechanisms.
8. **User Interface:** The portal should have an intuitive and responsive user interface that is easy to navigate for all stakeholders.

## **Performance Requirements**

The following are the performance requirements for the system:

1. **Event Management:** The system should be able to handle a minimum of 50 event proposals per day, with an average response time of less than 24 hours.
2. **Event Registration:** The system should be able to handle a minimum of 500 registrations per day for approved events, with an average processing time of less than 1 hour.
3. **Club Information Management:** The system should be able to handle a minimum of 50 club information updates per day, with an average response time of less than 24 hours.
4. **New Club Formation Requests:** The system should be able to handle a minimum of 5 new club formation requests per day, with an average processing time of less than 1 week.
5. **User Management:** The system should be able to handle a minimum of 50 user account modifications per day, with an average response time of less than 24 hours.
6. **Reporting:** The system should generate reports in real-time and provide them to the admin within 24 hours of request.
7. **Security:** The system should be able to handle a minimum of 50 login attempts per second, with an average response time of less than 1 second.
8. **User Interface:** The portal should load in less than 3 seconds on average for all stakeholders.

## **Design Constraints**

The following are the design constraints for the system:

1. **Technology Stack:** The system must be built using a web-based technology stack, such as HTML5, CSS3, and JavaScript.
2. **Integration:** The system should integrate with Google API for user authentication and authorization.
3. **Scalability:** The system should be designed to handle an increase in the number of users, events, and clubs over time.
4. **Security:** The system must implement appropriate security measures to protect user data from unauthorized access or breaches.
5. **User Experience:** The portal should provide a seamless user experience for all stakeholders, with an intuitive and responsive interface.
6. **Data Management:** The system should be able to store and manage large amounts of data related to events, clubs, users, and reports.
7. **Reporting:** The system should generate detailed reports on event attendance, club membership, and new club formation requests.
8. **Collaboration:** The system should enable collaboration between club coordinators, students, and the Student Council Clubs Coordinator through a centralized platform.

## External Interfaces

The following are the external interfaces for the system:

1. User Interface: The portal will be accessible to all stakeholders through a web-based interface that is responsive on various devices.
2. Google API: The system will integrate with Google API for user authentication and authorization.
3. Student Council Clubs Coordinator: The system should provide an easy-to-use interface for the Student Council Clubs Coordinator to review event proposals, approve or deny new club formation requests, and view reports on event attendance and club membership.
4. Admin: The system should provide a user-friendly interface for the admin to manage user accounts, access reports, and monitor system activity.
5. Reporting: The system will generate detailed reports on event attendance, club membership, and new club formation requests that can be accessed by all stakeholders through the web-based interface.

## Security Requirements

The following are the security requirements for the system:

1. Authentication: The system should implement user authentication mechanisms to ensure only authorized users have access to sensitive data and features.
2. Authorization: The system should provide role-based authorization to ensure that each stakeholder has access only to their designated areas of the portal.
3. Data Encryption: The system should encrypt all sensitive data, such as passwords and credit card information, to protect it from unauthorized access or breaches.
4. Access Control: The system should implement appropriate access control measures to restrict access to sensitive data and features based on user roles and permissions.
5. Input Validation: The system should validate all user input to prevent common attacks such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
6. Output Encoding: The system should encode all output data to protect against XSS attacks.
7. Error Handling: The system should handle errors gracefully and provide appropriate error messages to users.
8. Logging: The system should implement logging mechanisms to track user activity, system events, and security-related incidents.
9. Compliance: The system should comply with relevant laws and regulations related to data privacy and security.

## Use Cases for the Application

The following are some use cases for the application:

1. Club Coordinator Submits an Event Proposal \* A club coordinator submits an event proposal through the portal, including details such as date, time, location, and purpose of the event. \* The Student Council Clubs Coordinator reviews the proposal and provides approval or denial based on availability of resources and impact on other clubs.
2. Student Registers for an Approved Event \* A student logs in to the portal using their college email address through Google API. \* They view a list of approved events and select one to register for it.

3. User Requests New Club Formation \* A user submits a request for the formation of a new club through the portal, including details such as club name, purpose, proposed meeting time, location, and list of proposed members. \* The Student Council Clubs Coordinator reviews the request and provides approval or denial based on availability of resources and impact on other clubs.
4. Admin Manages User Accounts \* An admin logs in to the portal using their credentials. \* They view a list of user accounts and perform actions such as adding new users, modifying existing user details, and deleting inactive users.
5. Reporting: Event Attendance \* The system generates reports on event attendance for club coordinators, students, and the Student Council Clubs Coordinator.
6. Reporting: New Club Formation Requests \* The system generates reports on new club formation requests for club coordinators, students, and the Student Council Clubs Coordinator.
7. Reporting: Club Membership \* The system generates reports on club membership for club coordinators, students, and the Student Council Clubs Coordinator.

## Glossary of Terms

The following are some terms used in the SRS document:

1. Club Coordinator: An individual who organizes events for their respective clubs.
2. Event Proposal: A request submitted by a club coordinator to organize an event, including details such as date, time, location, and purpose of the event.
3. Student Council Clubs Coordinator: A member of the Student Council responsible for approving or denying event proposals submitted by club coordinators.
4. User Account: An account created in the system to enable access to the portal by stakeholders such as club coordinators, students, and the Student Council Clubs Coordinator.
5. Event Registration: The process of registering for an approved event through the portal using a college email address.
6. Reporting: Generating reports on various aspects of the system, such as event attendance, new club formation requests, and club membership.

**Can you use this format for the use cases?**

**Use case**

- Actor
- Purpose
- Event Flow
- Special/Exceptional Requirements

**Please use this and rewrite the SRS.**

Sure, here is a revised version of the SRS document using the provided format for the use cases:

# Software Requirements Specification

## (SRS) Document for College Club

# Management Portal

## Introduction

The following document outlines the requirements for a college club management portal, which will provide a medium for club coordinators to get approval to organize events and allow students to register for approved events. The system will also enable users to access basic information about clubs and their members, as well as request the formation of new clubs.

## Problem Background

The current process for club coordinators to organize events is manual and time-consuming. They must submit a written proposal to the Student Council Clubs Coordinator, who then reviews it and provides approval or denial based on the availability of resources and the impact on other clubs. The system will automate this process and provide a digital platform for club coordinators to manage their events and communicate with students interested in participating.

## Stakeholders / Users

- Club Coordinators: These are the individuals who organize events for their respective clubs. They will use the portal to submit event proposals, access information about clubs and members, and request the formation of new clubs.
- Students: These are the individuals who can register for approved events and view basic information about clubs.
- Student Council Clubs Coordinator: This is a member of the Student Council responsible for approving or denying event proposals submitted by club coordinators.
- Admin: This is an admin staff from the Student Affairs department of the university, who will have the right to change the club coordinators and manage user accounts.

## Functional Requirements

The following are the functional requirements for the system:

1. Event Management: The portal should allow club coordinators to submit event proposals, which include details such as date, time, location, and purpose of the event. The Student Council Clubs Coordinator must be able to review these proposals and provide approval or denial based on availability of resources and impact on other clubs.
2. Event Registration: Once an event is approved, students should be able to register for it through the portal using their college email addresses through Google API.
3. Club Information Management: The portal should allow users to access basic information about clubs and their members. This includes club name, purpose, meeting time, location, and contact details of club coordinators.
4. New Club Formation Requests: Users should be able to request the formation of a new club through the portal. This will involve providing details such as club name, purpose, proposed meeting time, location, and list of proposed members. The Student Council Clubs Coordinator must review these requests and provide approval or denial based on availability of resources and impact on other clubs.
5. User Management: The system should allow the admin to manage user accounts for club

coordinators, students, and the Student Council Clubs Coordinator. This includes adding new users, modifying existing user details, and deleting inactive users.

6. Reporting: The system should provide reporting features that enable the admin to view statistics on event attendance, club membership, and new club formation requests.
7. Security: The system must ensure the privacy and security of user data by implementing appropriate authentication and authorization mechanisms.
8. User Interface: The portal should have an intuitive and responsive user interface that is easy to navigate for all stakeholders.

## Performance Requirements

The following are the performance requirements for the system:

1. Event Management: The system should be able to handle a minimum of 50 event proposals per day, with an average response time of less than 24 hours.
2. Event Registration: The system should be able to handle a minimum of 500 registrations per day for approved events, with an average processing time of less than 1 hour.
3. Club Information Management: The system should be able to handle a minimum of 50 club information updates per day, with an average response time of less than 24 hours.
4. New Club Formation Requests: The system should be able to handle a minimum of 5 new club formation requests per day, with an average processing time of less than 1 week.
5. User Management: The system should be able to handle a minimum of 50 user account modifications per day, with an average response time of less than 24 hours.
6. Reporting: The system should generate reports in real-time and provide them to the admin within 24 hours of request.
7. Security: The system must implement appropriate security measures to protect user data from unauthorized access or breaches.
8. User Interface: The portal should load in less than 3 seconds on average for all stakeholders.

## Design Constraints

The following are the design constraints for the system:

1. Technology Stack: The system must be built using a web-based technology stack, such as HTML5, CSS3, and JavaScript.
2. Integration: The system should integrate with Google API for user authentication and authorization.
3. Scalability: The system should be designed to handle an increase in the number of users, events, and clubs over time.
4. Security: The system must implement appropriate security measures to protect user data from unauthorized access or breaches.
5. User Experience: The portal should provide a seamless user experience for all stakeholders, with an intuitive and responsive interface.
6. Data Management: The system should be able to store and manage large amounts of data related to events, clubs, users, and reports.
7. Reporting: The system should generate detailed reports on event attendance, club membership, and new club formation requests that can be accessed by all stakeholders through the web-based interface.
8. Collaboration: The system should enable collaboration between club coordinators, students, and the Student Council Clubs Coordinator through a centralized platform.

## External Interfaces

The following are the external interfaces for the system:

1. User Interface: The portal will be accessible to all stakeholders through a web-based interface that is responsive on various devices.
2. Google API: The system will integrate with Google API for user authentication and authorization.
3. Student Council Clubs Coordinator: The system should provide an easy-to-use interface for the Student Council Clubs Coordinator to review event proposals, approve or deny new club formation requests, and view reports on event attendance and club membership.
4. Admin: The system should provide a user-friendly interface for the admin to manage user accounts, access reports, and monitor system activity.
5. Reporting: The system will generate detailed reports on event attendance, club membership, and new club formation requests that can be accessed by all stakeholders through the web-based interface.

## Security Requirements

The following are the security requirements for the system:

1. Authentication: The system should implement user authentication mechanisms to ensure only authorized users have access to sensitive data and features.
2. Authorization: The system should provide role-based authorization to ensure that each stakeholder has access only to their designated areas of the portal.
3. Data Encryption: The system should encrypt all sensitive data, such as passwords and credit card information, to protect it from unauthorized access or breaches.
4. Access Control: The system should implement appropriate access control measures to restrict access to sensitive data and features based on user roles and permissions.
5. Input Validation: The system should validate all user input to prevent common attacks such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
6. Output Encoding: The system should encode all output data to protect against XSS attacks.
7. Error Handling: The system should handle errors gracefully and provide appropriate error messages to users.
8. Logging: The system should implement logging mechanisms to track user activity, system events, and security-related incidents.
9. Compliance: The system should comply with relevant laws and regulations related to data privacy and security.

## Use Cases for the Application

The following are some use cases for the application:

1. Club Coordinator Submits an Event Proposal \* Actor: Club Coordinator \* Purpose: To submit an event proposal to the Student Council Clubs Coordinator for approval. \* Event Flow: The club coordinator logs in to the portal, fills out a form with details about the proposed event, and submits it to the system. The Student Council Clubs Coordinator reviews the proposal and provides approval or denial based on availability of resources and impact on other clubs. \* Special/Exceptional Requirements: The system should provide an easy-to-use interface for club coordinators, with clear instructions on how to

submit event proposals. The Student Council Clubs Coordinator must receive a notification when a new proposal is submitted, so they can review it in a timely manner.

2. Student Registers for an Approved Event \* Actor: Student \* Purpose: To register for an approved event through the portal using their college email address. \* Event Flow: The student logs in to the portal using their college email address through Google API, views a list of approved events, and selects one to register for it. The system validates their registration and provides confirmation of their participation. \* Special/Exceptional Requirements: The system should provide clear instructions on how students can access event information and register for events. Students must be able to view the status of their registration in real-time, so they know if they have been approved or not.
3. User Requests New Club Formation \* Actor: User (can be a club coordinator, student, or other stakeholder) \* Purpose: To request the formation of a new club through the portal. \* Event Flow: The user logs in to the portal, fills out a form with details about the proposed club, and submits it to the system. The Student Council Clubs Coordinator reviews the proposal and provides approval or denial based on availability of resources and impact on other clubs. \* Special/Exceptional Requirements: The system should provide an easy-to-use interface for users to request new club formation, with clear instructions on what information is required. The Student Council Clubs Coordinator must receive a notification when a new proposal is submitted, so they can review it in a timely manner.
4. Admin Manages User Accounts \* Actor: Admin \* Purpose: To manage user accounts for club coordinators, students, and the Student Council Clubs Coordinator through the portal. \* Event Flow: The admin logs in to the portal using their credentials, views a list of user accounts, and performs actions such as adding new users, modifying existing user details, and deleting inactive users. \* Special/Exceptional Requirements: The system should provide an easy-to-use interface for admins to manage user accounts, with clear instructions on what information is required. Admins must be able to view the status of each account in real-time, so they know if it has been approved or not.
5. Reporting: Event Attendance \* Actor: Admin \* Purpose: To generate reports on event attendance for club coordinators, students, and the Student Council Clubs Coordinator through the portal. \* Event Flow: The admin logs in to the portal using their credentials, selects a date range, and generates a report that shows the number of attendees for each approved event during that time period. \* Special/Exceptional Requirements: The system should provide an easy-to-use interface for admins to generate reports on event attendance, with clear instructions on what information is required. Admins must be able to view the status of each report in real-time, so they know if it has been generated or not.
6. Reporting: New Club Formation Requests \* Actor: Admin \* Purpose: To generate reports on new club formation requests for club coordinators, students, and the Student Council Clubs Coordinator through the portal. \* Event Flow: The admin logs in to the portal using their credentials, selects a date range, and generates a report that shows the number of new club formation requests submitted during that time period. \* Special/Exceptional Requirements: The system should provide an easy-to-use interface for admins to generate reports on new club formation requests, with clear instructions on what information is required. Admins must be able to view the status of each report in real-time, so they know if it has been generated or not.
7. Reporting: Club Membership \* Actor: Admin \* Purpose: To generate reports on club membership for club coordinators, students, and the Student Council Clubs Coordinator through the portal. \* Event Flow: The admin logs in to the portal using their credentials, selects a date range, and generates a report that shows the number of new members added to each club during that time period. \* Special/Exceptional Requirements: The system should provide an easy-to-use interface for admins to generate reports on club membership, with clear instructions on what information is required. Admins must be able to view the status of each report in real-time, so they know if it has been generated or not.

## Glossary of Terms

The following are some terms used in the SRS document:

1. Club Coordinator: An individual who organizes events for their respective clubs.
2. Event Proposal: A request submitted by a club coordinator to organize an event, including details such as date, time, location, and purpose of the event.
3. Student Council Clubs Coordinator: A member of the Student Council responsible for approving or denying event proposals submitted by club coordinators.
4. User Account: An account created in the system to enable access to the portal by stakeholders such as club coordinators, students, and the Student Council Clubs Coordinator.
5. Event Registration: The process of registering for an approved event through the portal using a college email address.
6. Reporting: Generating reports on various aspects of the system, such as event attendance, new club formation requests, and club membership.