Final Year Project

UML Class Diagram Compiler

FYP Group

Mehroz Ahmad 18F-0159
Wahaj Tahir 19F-1014
Ahmad Yasir 19F-0182

Supervised

By

Dr.Sajid Anwer

National University of Computer
and Emerging Sciences

Department of Computer Science
National University of Computer and Emerging Science
Chiniot Faisalabad Campus, Pakistan

2023

Contents

List of Tables

List of Figures

1 Introduction

UML class diagrams are essential for software development as they provide a visual representation of the structure and relationships among classes in a system. They aid in understanding the system's architecture, facilitating communication among stakeholders, and serving as a blueprint for implementation.

In many projects, identifying errors in class diagrams can be a challenging task.There is no definitive method for error detection in class diagrams, and the process can be time-consuming. Additionally, it can be difficult to recall previously made errors, leading to potential repetition of mistakes.

To address this challenge, we are embarking on the development of a web application called the "UML-Class Diagram Compiler."This project aims to assist individuals struggling with error identification in class diagrams. The application will take a diagram as input, analyze it based on the established model, and identify the syntactic and semantic errors within the class diagram. Moreover,it will maintain a record of identified errors to enhance the quality of artifacts and prevent their recurrence in the future.

2 Vision Document

In this section, the project vision is discussed in detail.

2.1 Problem Statement

Table 1: Problem Statement

Problem	**Description**
The problem	In many projects, identifying errors in class diagrams can be a challenging task.There is no definitive method for error detection in class diagrams, and the process can be time-consuming. Additionally, it can be difficult to recall previously made errors, leading to potential repetition of mistakes.
Affects	The people related to the IT industry i.e. students, internees, teachers, developers etc...
The result of which	They can find errors in class diagrams efficiently and they will also have a record of their past errors on the history page.
Benefits of	UML-Class Diagram compiler is the platform which is providing solution to syntax,semantic errors and also generating skeleton code.

2.2 Business Opportunity

We are living in an era, in which most people are earning from the IT industry. People are doing business by doing IT-related projects and making their side income. Also, developers are having problems, when they want to find errors in the class diagrams but they could not find errors efficiently, as most of the

time when there is an error in the class diagram and they start development, it leads to a big problem for the developer as well as investor. So, there is a huge demand for a platform, which resolves all these problems.

2.3 Objectives

- A system where IT-related people will be able to find errors in class diagrams efficiently without the help of an expert.
- A user can also view his/her past errors and improve artifacts in the future.

2.4 Scope

UML-Class Diagram compiler is a framework for automatic transformation of UML diagram into a string and then compilation of that string for syntactic correctness and verification, which is kind of semantic verification. The compiler will also produce the skeleton code of the given class diagram. The framework is divided into 3 parts.

1. **XML to string converter:** This module will convert the XML form of UML diagram into a string that will follow the context free grammar.
2. **UML compiler:** This module will check the syntactic correctness of the diagram and verification of consistency among the diagrams.
3. **Code generator:** This module will allow user to generate the skeleton code of the given class diagram.

2.5 Constraints

- The system shall not require any hardware development or procurement.
- The system shall take class diagram as an input in XML format.
- The system shall use SQL relational database.
- The system shall check the ambiguity and validation of class diagram.
- The system shall produce a skeleton code of the UML class diagram.
- If there is an internet problem, then might be simple web page will be loaded without graphics.
- If the amount of memory available in device is low, the system may ask your application to shut down or sacrifice cached data, slowing program execution.

2.6 Stakeholder and User Descriptions

Stakeholder and User Descriptions are defined in detail below.

2.6.1 Market Demographics

Initially, our target market is Pakistan to provide a better platform for developers to find errors in class diagrams efficiently. The IT industry is growing day by day and it has much potential. To overcome this issue, we aim to design a “UML-Class Diagram compiler”, which is a dedicated website for developers.

2.6.2 Stakeholder Summary

Table 2: Stakeholder Summary

Name	Description	Responsibilities
Class Diagram Developer	It includes those who are involved in designing and implementing this project.	**1.** Should design website considering users' needs. **2.** Proper testing should be applied. **3.** Should be deployed properly.
End-users	End users include all the people related to IT from anywhere in Pakistan.	**1.** Should know using the website
Supervisor of project	Supervisor will be stakeholders of this product as they are involved in the development process of this project with the development team.	**1.** Gives direction to the development team. **2.** Ensures that the system will be maintainable. **3.** Ensures that there will be market demand for the product's features. **4.** Monitors the project's progress. **5.** Ensures that the project complies with the documents generated during project planning and that work products are properly delivered.

2.6.3 User Environment

Web-based UML class diagram compiler typically includes the following components:

1. **Web Browser:**The compiler is accessed through a web browser, which must be compatible with the compiler's interface and features. Popular web browsers used for web-based UML class diagram compilers include Google Chrome, Mozilla Firefox, and Microsoft Edge.

2. **Internet Connection:** Since the compiler is web-based, it requires a stable internet connection to access and use the tool.

3. **Operating System:**The operating system used by the developer is not as important for a web-based compiler since the tool is accessed through the browser. However, the browser must be compatible with the developer's operating system.

4. **Hardware:**The computer hardware used by developers should meet the minimum requirements specified by the compiler, such as processor speed, memory, and storage space.

5. **Development Environment:**Depending on the web-based compiler, it may need to be integrated with the developer's preferred development environment, such as Eclipse, Visual Studio, or IntelliJ IDEA.

2.6.4 Stakeholder Profiles

The Stakeholder Profiles Attributes of the system are listed in this section.

2.6.4.1 Supervisors of the Project

Table 3: Supervisors of the project

Representatives	**Supervisor:** Dr.Sajid Anwer **Co-supervisor:** Mr.Asif Ameer
Description	They are involved in supervising the activities of the development process.
Type	They are technical stakeholders. They have expertise in domains that are being applied to this project.
Responsibilities	**1.**Gives direction to the development team. **2.**Ensures that the system will be maintainable. **3.**Ensures that there will be a market demand for the product's features. **4.**Monitor the project's progress. **5.**Ensures that the project complies with the documents generated during project planning and that work products are properly delivered. **6.**They will facilitate the development team to complete this project within specified resources.
Success Criteria	The completion of the feature being committed by the development team at the start of the project.
Involvement	**1.**Requirement reviewer **2.**Senior managers **3.**Reviews implementation

2.6.4.2 Development Team of the Project

Table 4: Development Team of the Project

Representative	**Mehroz Ahmad** **Wahaj Tahir** **Ahmad Yasir**
Description	They are involved in the development process of this project with the development team.
Type	They are technical stakeholders.
Responsibilities	**1.** Should design this website considering user's needs. **2.** Proper testing should be applied to make it as effective as possible.
Success Criteria	The completion of features that are being committed by the development team at the start of the project.
Involvement	**1.** Designers **2.** Testers
Deliverable	**1.** Documentation **2.** Implementation **3.** Data acquisition **4.** System Training

3 System Requirement Specification

In this section, the features and requirements of the system are explained.

3.1 System Features

1. Person login
2. Take XML format input
3. Convert String
4. Check the syntactic error
5. Check the semantic error
6. Save error
7. Generate skeleton code

3.2 Functional Requirements

Table 5: Functional Requirements

Functionality No.	Description
FR-1	This application will provide a dedicated page where users can either log in to their existing accounts or sign up for new ones.
FR-2	This application will accept input in the form of XML format files.
FR-3	This application will transform the XML file into a string format.
FR-4	This application will incorporate context-free grammar rules..
FR-5	Using context-free grammar rules, this application will verify and identify syntax and semantic errors in the provided input.
FR-6	This application will automatically generate a skeleton code structure of provided input.
FR-7	This application will include a history page that will maintain a record of the user's previous mistakes for reference and review.
FR-8	The administrator will have the ability to add words in natural language to the system.
FR-9	The administrator will have the capability to define the natural language using context-free grammar in the system.
FR-10	The administrator will have the authority to implement and configure the rules in the system.

3.3 Non-Functional Requirements

Table 6: Non-Functional Requirements

Non-Functionality	Description
NFR-1	The application should be active to response 24 hours a day and 7 days a week.
NFR-2	The UML class diagram compiler should provide a user-friendly interface with clear instructions and intuitive navigation. • The application should have a user satisfaction rating of at least 60% based on user surveys or feedback. • The average time for users to learn and become proficient in using the compiler should be less than 1 hour.
NFR-3	The average time to fix bugs or address issues reported by users should be less than 3 days.
NFR-4	The UML class diagram compiler should have a low error rate, ensuring that it correctly identifies syntax and semantic errors in class diagrams at least 75% of the time.
NFR-5	The UML class diagram compiler should be able to handle an increasing number of class diagrams and users without significant degradation in performance. It should support a concurrent user load of at least 5 users.
NFR-6	The application should have a response time of less than 1.5 milliseconds for user interactions such as login, sign-up, or diagram validation.
NFR-7	The application should undergo regular security assessments and vulnerability testing to ensure the integrity of the system.

4 Use Case Diagram

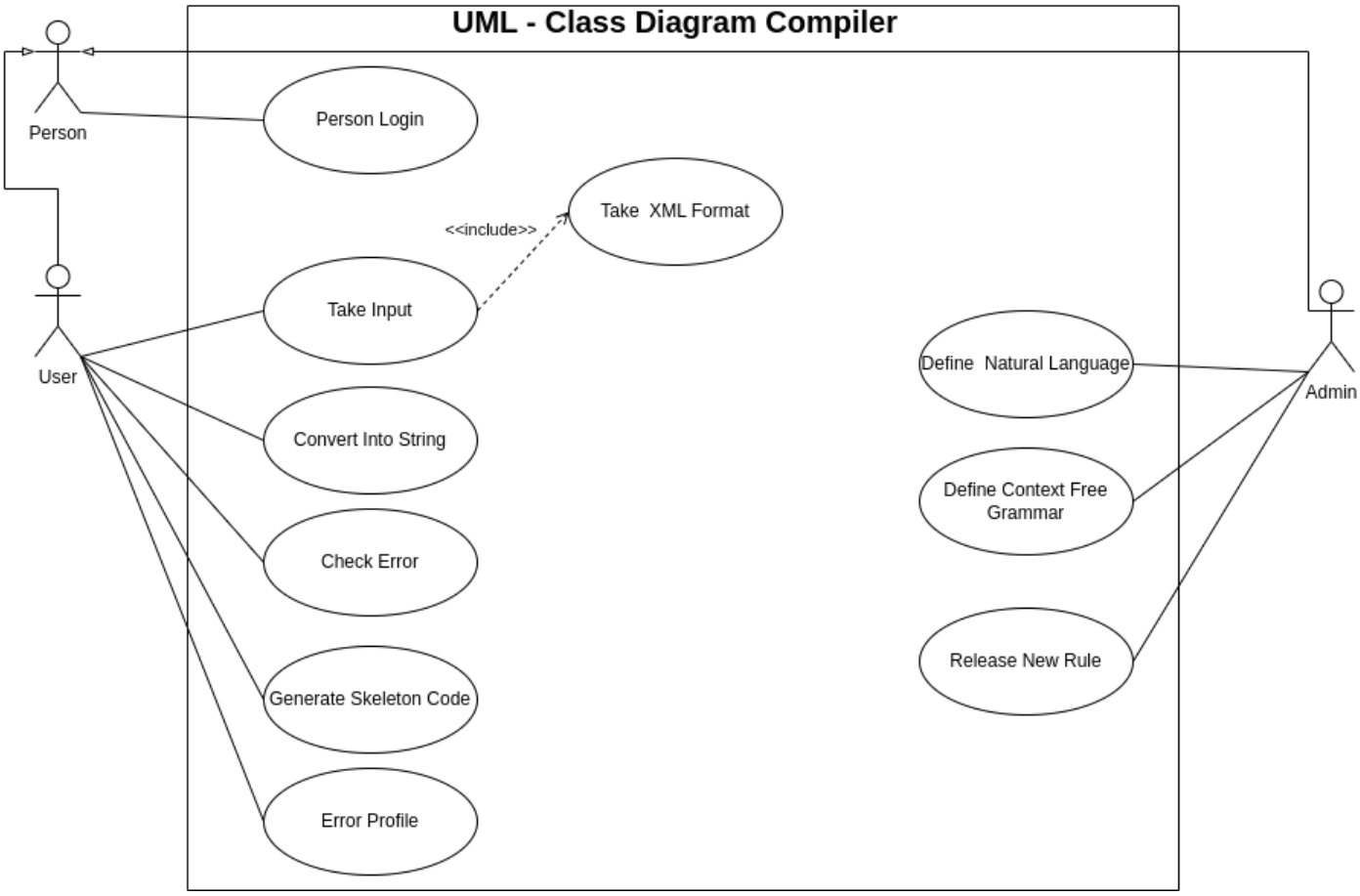

Figure 1: Use case Diagram

5 Expended use cases

5.1 Person Login

Table 7: ECU-01 Person Login

Use case ID	01
Use case Name	Person Login
Actor	Person
Description	To use the application, the user must create an account. If he already has the account, he will log in, and to manage the application, the admin needs an account by signing up, and if he already has one, sign in.
Trigger	The user has requested to login.
Pre-Condition	• User/Admin has an active internet connection. • User/Admin has a web browser • User/Admin will go to the website • User/Admin will type the credentials
Post-Condition	• Login successfully
Normal Flow	**1.**Users/Admins access the UML class diagram compiler web application through a web browser or dedicated client interface. **2.**Existing users navigate to the login page.Users/Admins enter their registered username/email and password in the provided fields. **3.**New users navigate to the signup page.Users enter their desired username, email, and password in the respective fields. **4.**After login or signup, the system redirects the user to their dashboard or main page.
Alternate Flow	**2.1.**If the user enters incorrect login credentials, such as an invalid username or password, the system displays an error message indicating an unsuccessful login attempt. **3.1.**During the signup process, the system may check for duplicate accounts with the same email or username. **4.1.**After resolving these issues the system redirects the user to their dashboard or main page.
Special Requirement	Availability of Internet
Frequency of Use	High
Assumption	The user knows how to log in.

5.2 Take Input

Table 8: ECU-02 Take Input

Use case ID	02
Use case Name	Take Input
Actor	User
Description	The user would have to give input to the system in an XML format which will proceed the system further.
Trigger	The user has clicked to give the upload file.
Pre-Condition	• The user has logged in successfully. • User must be in the application
Post-Condition	• The given file will be uploaded for evaluation.
Normal Flow	**1.**The user creates or obtains a valid XML file containing the UML class diagram representation. **2.**Within the application, there is an option to select or upload the input file. **3.**The UML class diagram compiler parses and reads the content of the input file.
Alternate Flow	**1.1.**If the user attempts to upload a file that does not adhere to the required format.It will throw an error. **2.1.**The user is informed about the missing or incomplete data and may be prompted to provide the necessary information or correct the existing data. **3.1.**In the case of encountering unrecognized elements or unsupported UML constructs within the input file, the compiler raises warnings.
Special Requirement	Availability of file in XML format
Frequency of Use	High
Assumption	The user knows how to upload a file

5.3 Take XML Format

Table 9: ECU-03 Take XML Format

Use case ID	03
Use case Name	Take XML Format
Actor	User
Description	The system will take the XML format, convert it into a string, and perform error-checking functionalities.
Trigger	Giving input to the system
Pre-Condition	• User must be in the system • The file must be in XML format
Post-Condition	• File is uploaded successfully
Normal Flow	**1.**The user creates or obtains a valid XML file containing the UML class diagram representation. **2.**Within the application, there is an option to select or upload the input file. **3.**The UML class diagram compiler parses and reads the content of the input file.
Alternate Flow	**1.1.**If the user attempts to upload a file that does not adhere to the required format.It will throw an error. **2.1.**The user is informed about the missing or incomplete data and may be prompted to provide the necessary information or correct the existing data. **3.1.**In the case of encountering unrecognized elements or unsupported UML constructs within the input file, the compiler raises warnings.
Special Requirement	The user should click on the input icon
Frequency of Use	High
Assumption	The user should know how to give input in XML format.

5.4 Convert Into String

Table 10: ECU-04 Convert Into String

Use case ID	05
Use case Name	Convert String
Actor	User
Description	After giving the input the system will convert the file into a string and make it readable.
Trigger	The user will give input to the system.
Pre-Condition	• The user must be in the system • The file format should be XML
Post-Condition	• Convert into string successfully
Normal Flow	**1.**The user provides an XML file containing the UML class diagram. **2.**The UML class diagram compiler parses the XML file, extracting the necessary class diagram information. **3.**The string representation may include the class definitions, relationships, attributes, and methods in a readable format.
Alternate Flow	**1.1.**An error message is displayed, indicating the invalid XML format, and the user is prompted to provide a valid XML file in order to make it readable. **2.1.**The compiler raises warnings or errors, indicating the missing or incomplete data, and the user is informed about the specific elements that need to be provided or corrected. **3.1.**If there are issues during the conversion process, such as unexpected data or errors in the parsed XML, the compiler may encounter difficulties in converting the XML to a string representation.
Special Requirement	Specific format input
Frequency of Use	High
Assumption	The user should know how to upload a file.

5.5 Check Error

Table 11: ECU-05 Check Error

Use case ID	04
Use case Name	Check error
Actor	User
Description	The system would find all possible syntactic errors and display those errors on screen.
Trigger	The user has clicked the icon to check the syntactic error
Pre-Condition	• File must be uploaded successfully
Post-Condition	• The system will check and display all possible syntactic and semantic errors in the class diagram.
Normal Flow	**1.**It checks if the elements are properly defined, the relationships are correctly specified, and the syntax rules of UML class diagrams are followed then it follows the instruction semantic rules. **2.**During the syntax or semantic error analysis, if any syntax or semantic errors are found, such as missing or misplaced brackets, incorrect notation usage, or invalid element declarations, the compiler identifies them. **3.**The UML class diagram compiler presents the syntax and semantic error messages or notifications to the user.
Alternate Flow	**1.1.**The compiler raises warnings or errors, indicating the presence of unsupported notations or constructs, and advises the user to remove or replace them with valid UML syntax. **2.2.**If the class diagram has incomplete or partially defined syntax, such as missing element declarations or unfinished relationships, the compiler recognizes the incomplete sections and semantic error can not be resolved.
Special Requirement	Availability of input
Frequency of Use	High
Assumption	The user knows the errors in the class diagram already

5.6 Generate Skeleton Code

Table 12: ECU-06 Generate Skeleton Code

Use case ID	07
Use case Name	Generate Skeleton Code
Actor	User
Description	After converting the input into the string, the user can generate its skeleton code by just clicking on generate skeleton code.
Trigger	The user will click on generate skeleton code
Pre-Condition	The file is uploaded in XML format successfully File is converted into string
Post-Condition	Skeleton code is generated
Normal Flow	**1.**Based on the information gathered, the compiler generates skeleton code that represents the structure and basic functionality of the classes in the diagram. **2.**The skeleton code may include class definitions, method signatures, variable declarations, and other basic code structures.
Alternate Flow	**1.1.**In some cases, the class diagram may have ambiguous or conflicting elements that can lead to ambiguous or conflicting code generation. **2.2.**The compiler generates an error message indicating the code generation failure and provides guidance to the user for troubleshooting or resolving the issue.
Special Requirement	User should give some input to check errors.
Frequency of Use	High
Assumption	Admin should know basic rules of the class diagram

5.7 Error Profile

Table 13: ECU-07 Error Profile

Use case ID	08
Use case Name	Error Profile
Actor	User
Description	The user's work will save on the history page and the user can review his/her errors. It is used for just keeping track of users' past mistakes.
Trigger	Click on save error button
Pre-Condition	• The user will find errors in the class diagram.
Post-Condition	• The errors of class diagram will be saved in his/her profile. • The user can also view his/her errors in the class diagram.
Normal Flow	**1.**The user interacts with the UML class diagram compiler, performing various actions such as uploading class diagrams, analyzing code, generating output, etc. **2.**The UML class diagram compiler records the user's activities, capturing relevant information such as the actions performed, timestamps, user identification, and any relevant data associated with the activities. **3.**The compiler stores the activity logs in a database which can be review later by users.
Alternate Flow	**1.**The compiler may raise warnings or errors, notifying the user about the missing or incomplete logs and suggesting possible actions to rectify the situation. **2.1.**The compiler generates error messages or notifications, informing the user about the logging failures and advising them to retry the action or report the issue to the system administrator.
Special Requirement	There should be some error found in the class diagram.
Frequency of Use	High
Assumption	The user should know how to find errors and save errors.

5.8 Define Natural Language

Table 14: ECU-08 Define Natural Language

Use case ID	09
Use case Name	Define Natural Language
Actor	Admin
Description	When an admin wants to add some new sentences to the system. He will click on the add sentence button and type the sentence he/she wants.
Trigger	Click on add sentence button
Pre-Condition	• Admin must be connected to the internet. • Admin must be logged into the system.
Post-Condition	• Arrange sentence according to the CFG rule.
Normal Flow	**1.**The admin type a set of natural language words or phrases that they want to associate with specific elements or concepts in the UML class diagram compiler. **2.**The admin will type the set of words or sentence and then add them.
Alternate Flow	**1.1.**Sentences starting with numbers or special characters generate an error. **2.1.**Add a sentence by clicking on the button
Special Requirement	Admin must be logged into system.
Frequency of Use	High
Assumption	The admin should know the rules.

5.9 Define Context Free Grammar

Table 15: ECU-9 Define Context Free Grammar

Use case ID	10
Use case Name	Define Context Free Grammar
Actor	Admin
Description	The admin who right the sentence must it according to the CFG rules, to make it understandable by the system by clicking on the define in CFG rule button will make the sentence according to the CFG rules.
Trigger	The user will click on the define in CFG rule button.
Pre-Condition	• It should be in a string
Post-Condition	• Implement the rule
Normal Flow	**1.**Type the sentence according to the context-free grammar. **2.**The sentence is arranged in context-free grammar rule successfully
Alternate Flow	**1.1.**First correct the ambiguity in the sentence. **2.1.**The sentence is arranged in context-free grammar rule successfully
Special Requirement	Rules should be written in natural language
Frequency of Use	High
Assumption	Admin should know how to convert rules in CFG

5.10 Release New Rule

Table 16: ECU-10 Release New Rule

Use case ID	11
Use case Name	Release New Rule
Actor	Admin
Description	The sentence which is arranged and correct according to the CFG rule will now be implemented in the system.
Trigger	Click on implement new rule
Pre-Condition	• The sentence arranged into CFG rule
Post-Condition	• Sentence is implemented successfully
Normal Flow	**1.**After converting the sentence according to context-free grammar, the admin will add the new rule. **2.**Sentence is implemented successfully
Alternate Flow	**1.1.**Sentence containing ambiguity, correct it **2.1.**Sentence is implemented successfully
Special Requirement	Admin should have written rules in form of CFG
Frequency of Use	High
Assumption	Admin should know whether rules are related to our problem or not

6 System Sequence Diagram

6.1 Login and input

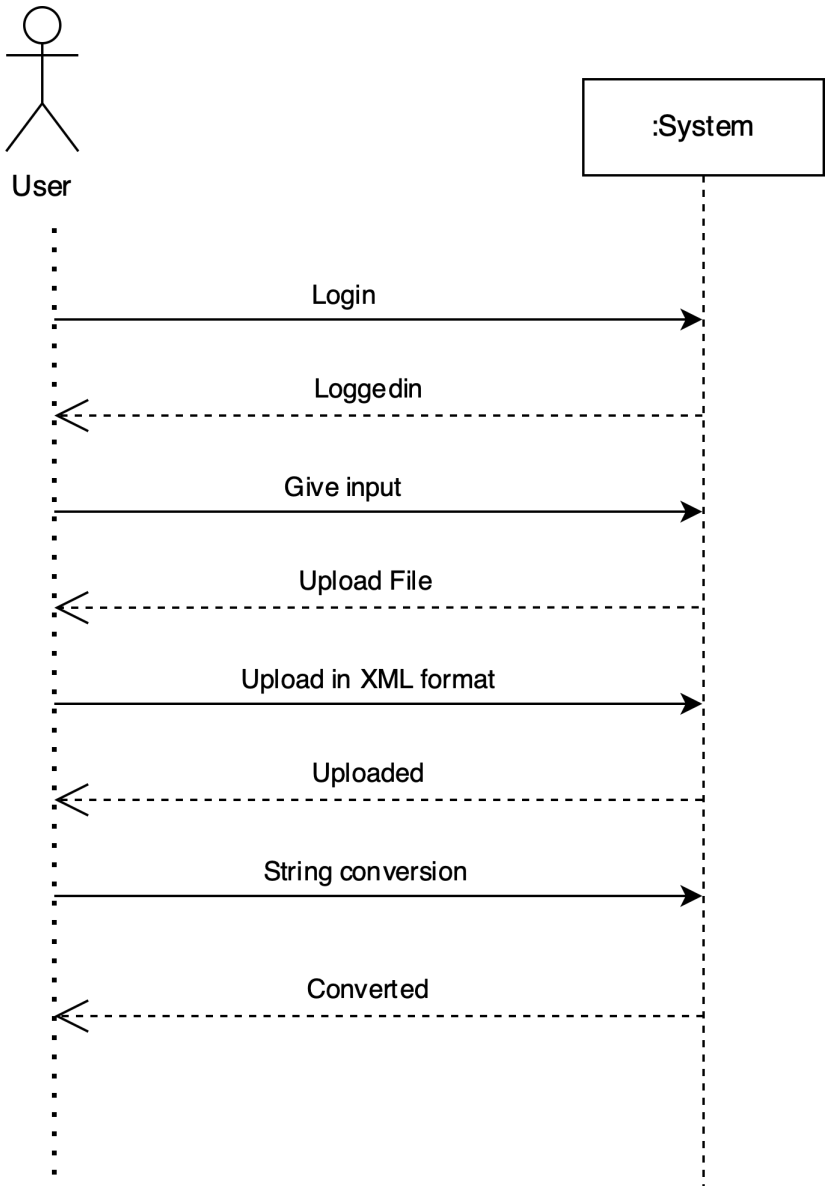

Figure 2: SSD-01 Login and input

6.2 Check Errors

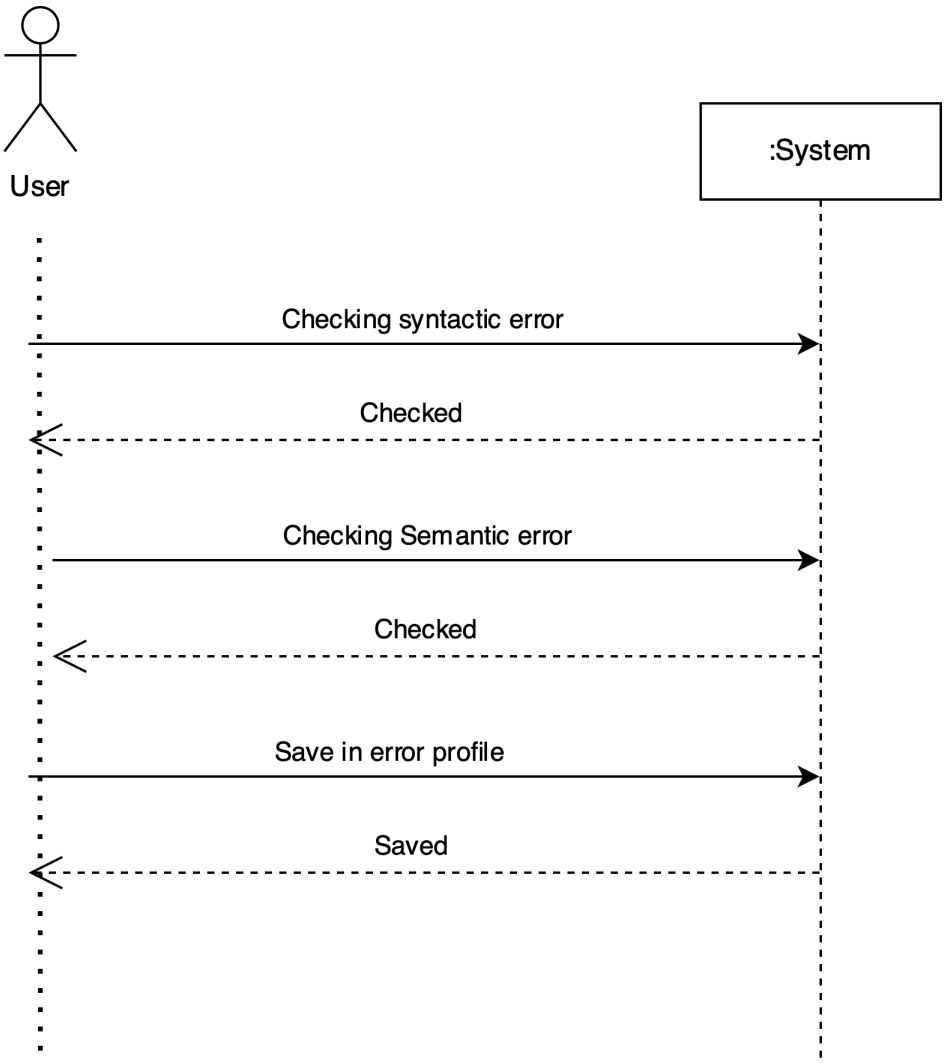

Figure 3: SSD-02 Check Error

6.3 Admin

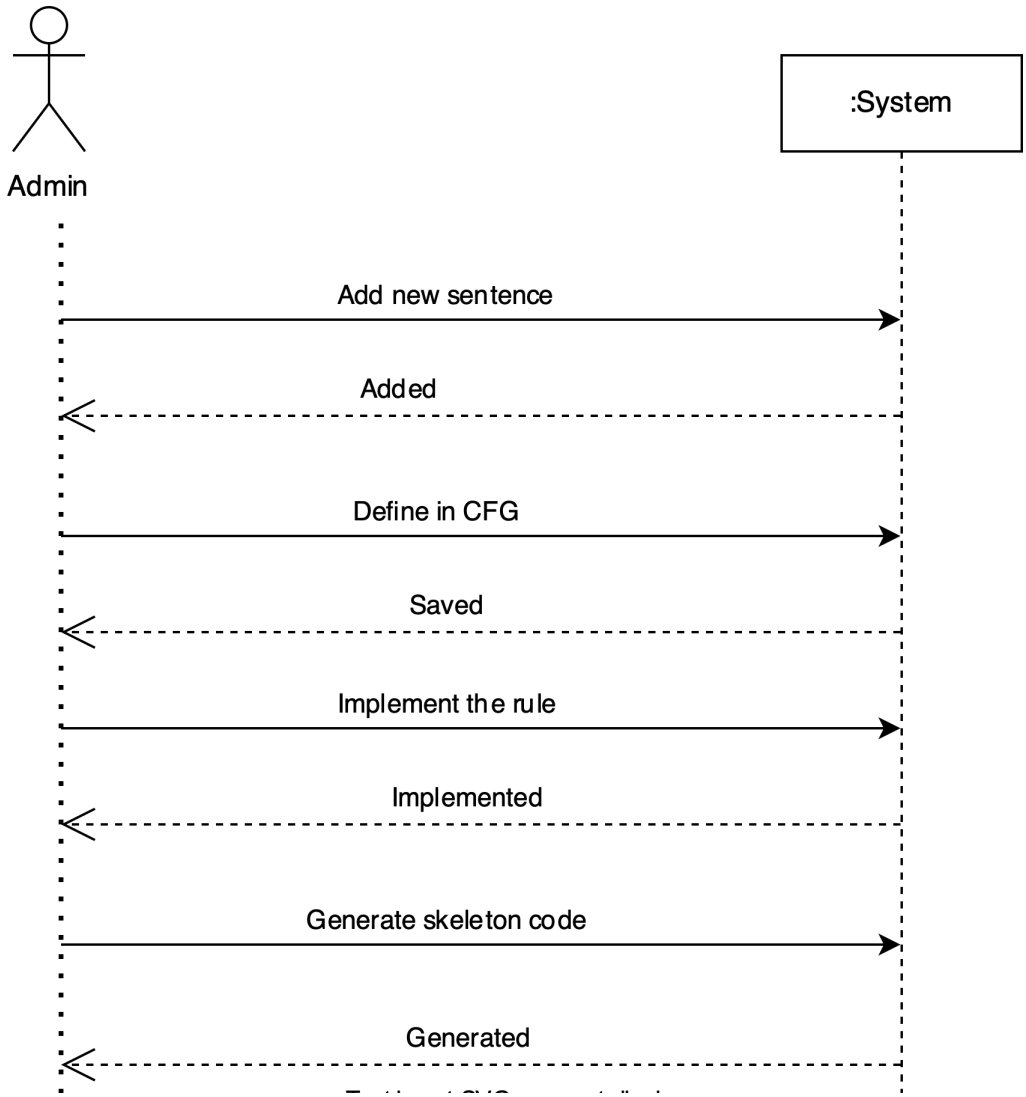

Figure 4: SSD-03 Admin

7 Activity Diagram

7.1 Admin

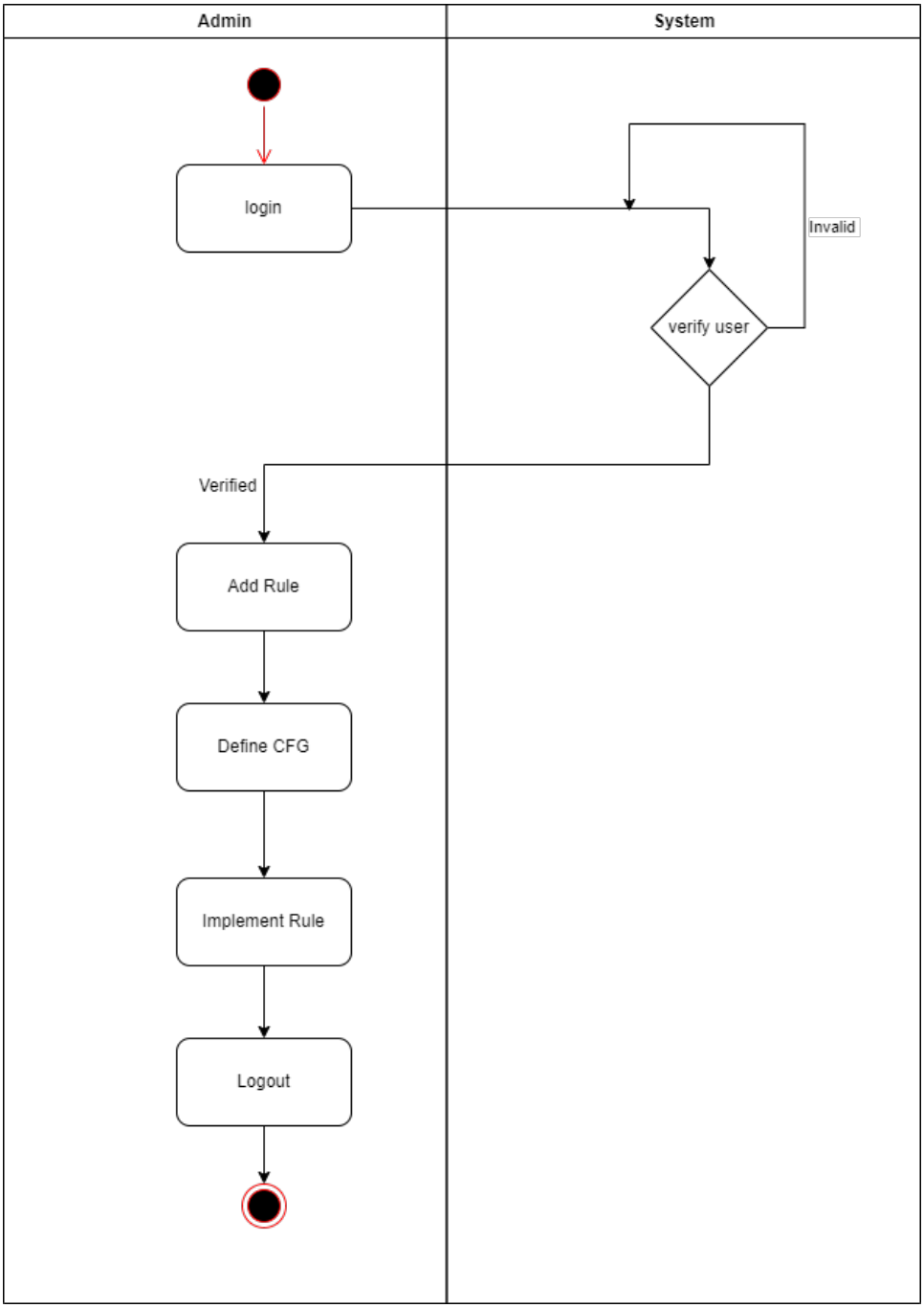

Figure 5: AD-01 Admin

7.2 User Input

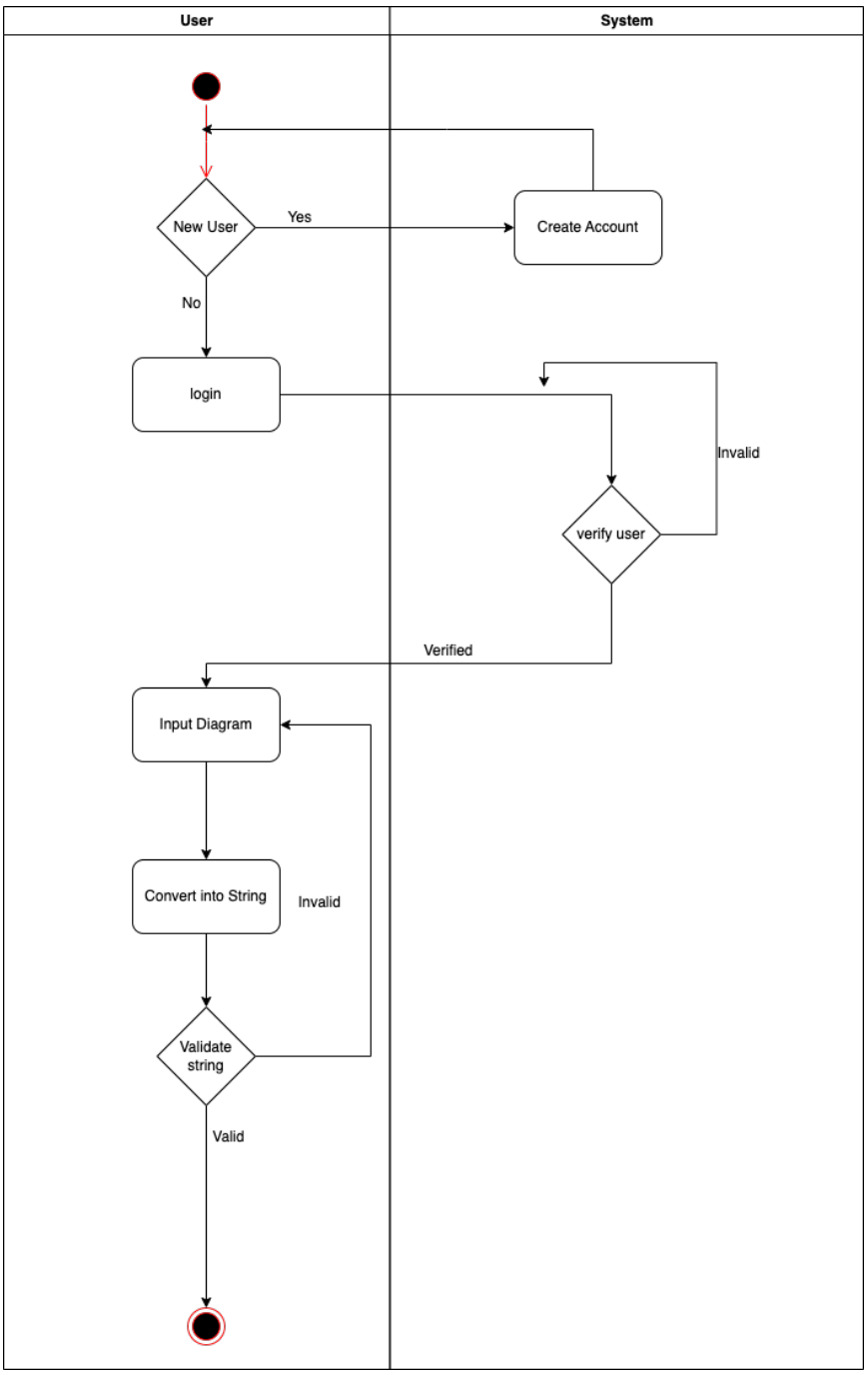

Figure 6: AD-02 User Input

7.3 Check Error

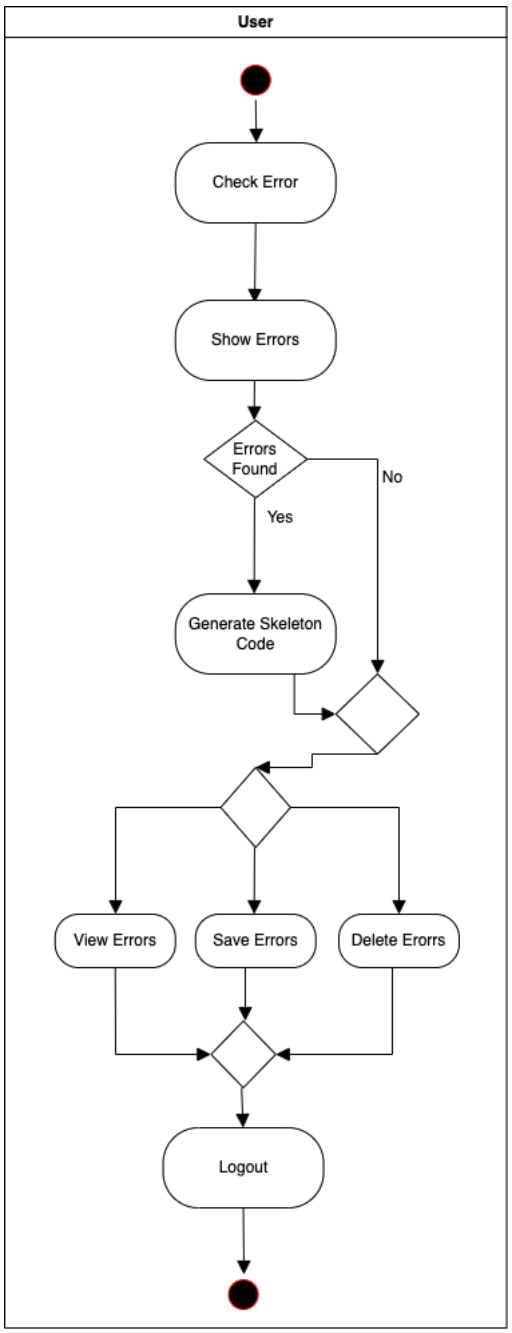

Figure 7: AD-03 Check Error

8 Domain Model

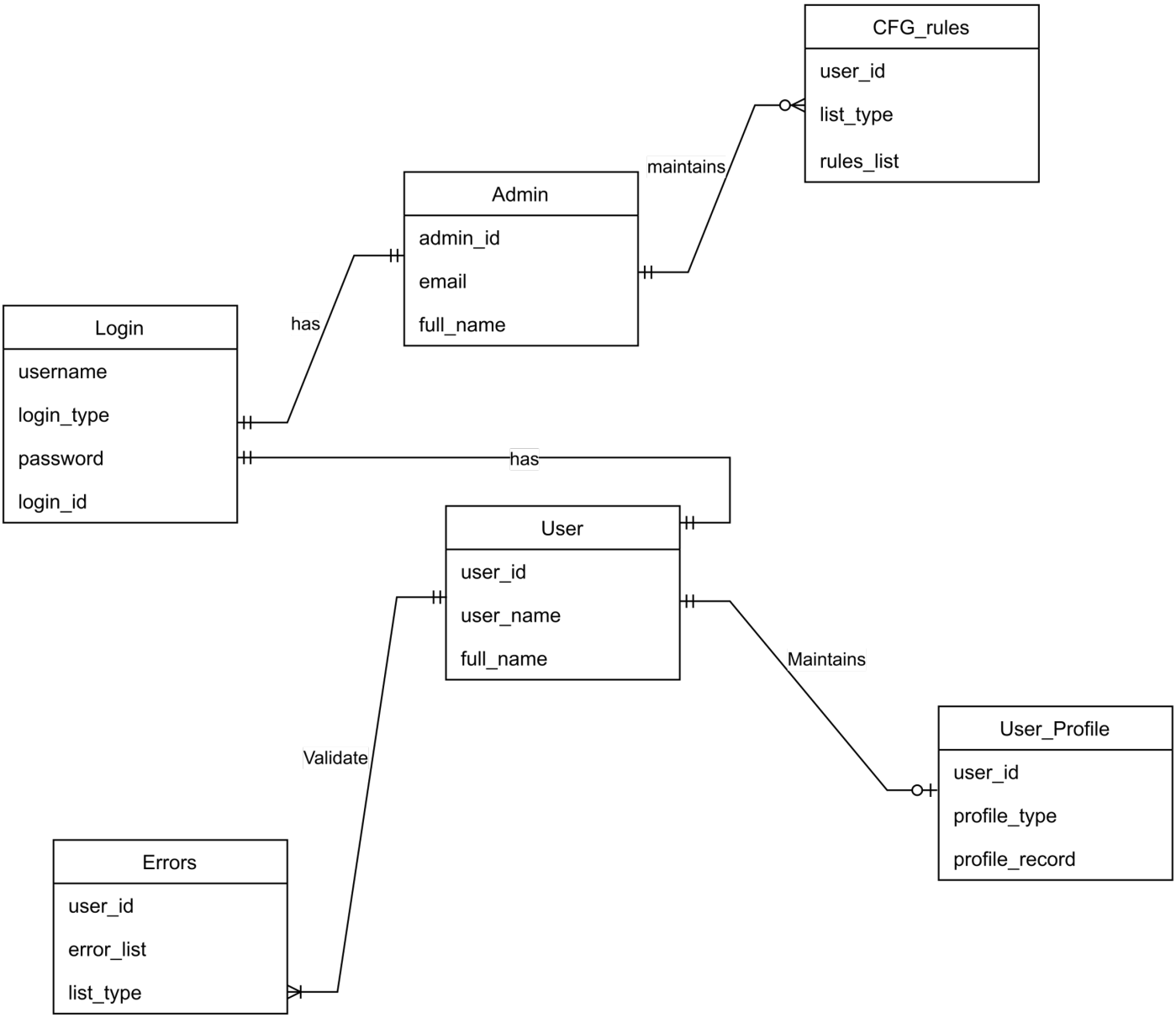

Figure 8: Domain Model

9 Class Diagram

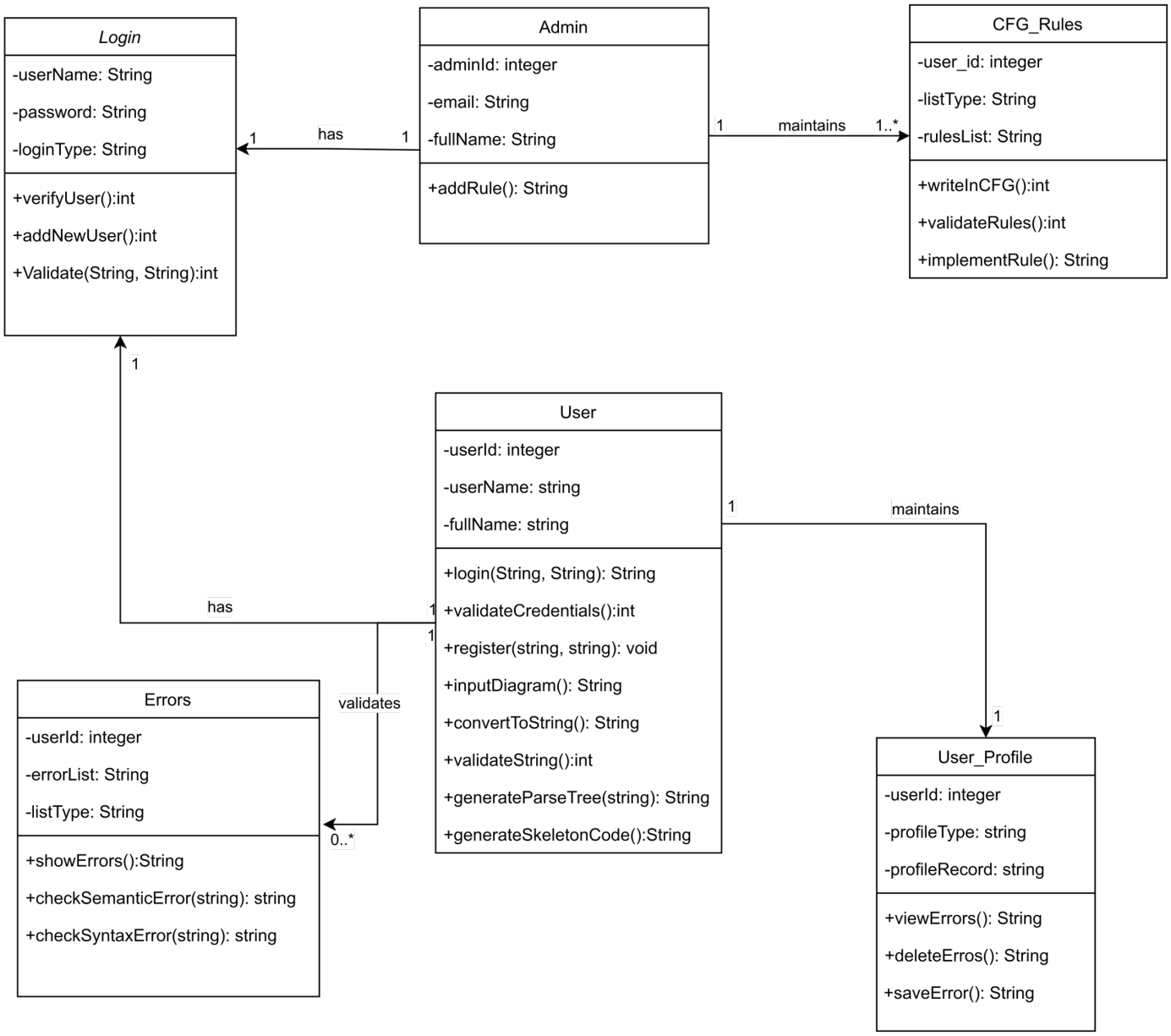

Figure 9: Class Diagram

10 Sequence Diagram

10.1 Person Sign-up

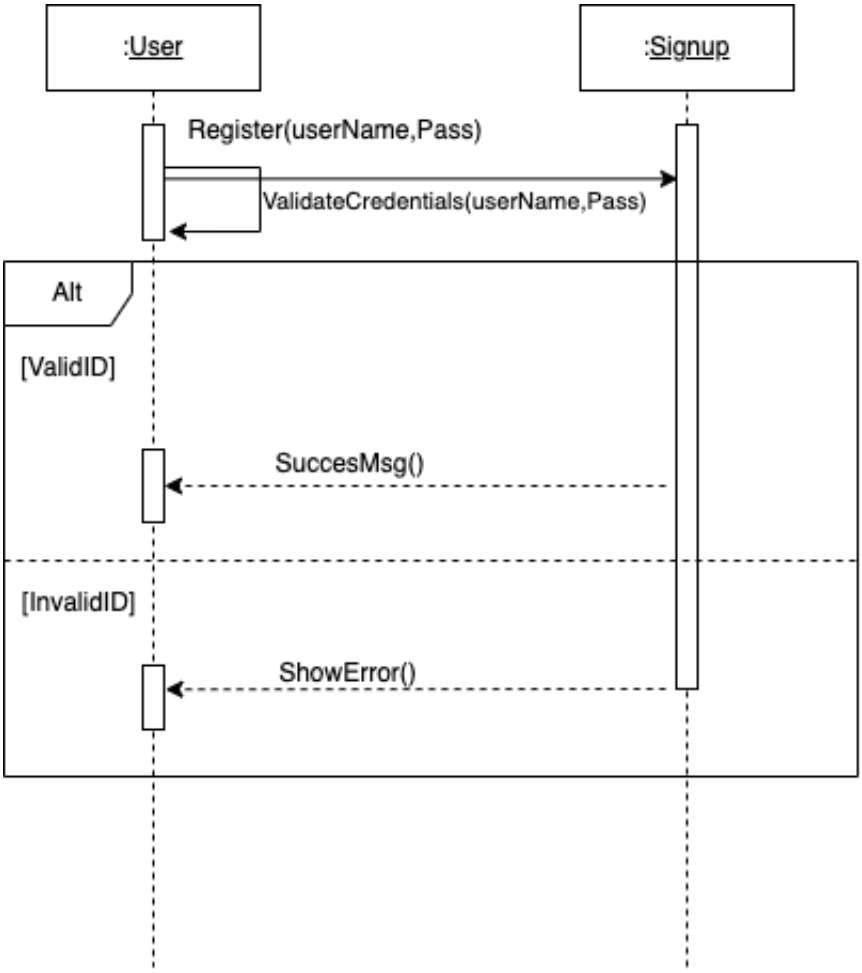

Figure 10: SD-01 Person Sign-up

10.2 Person Login

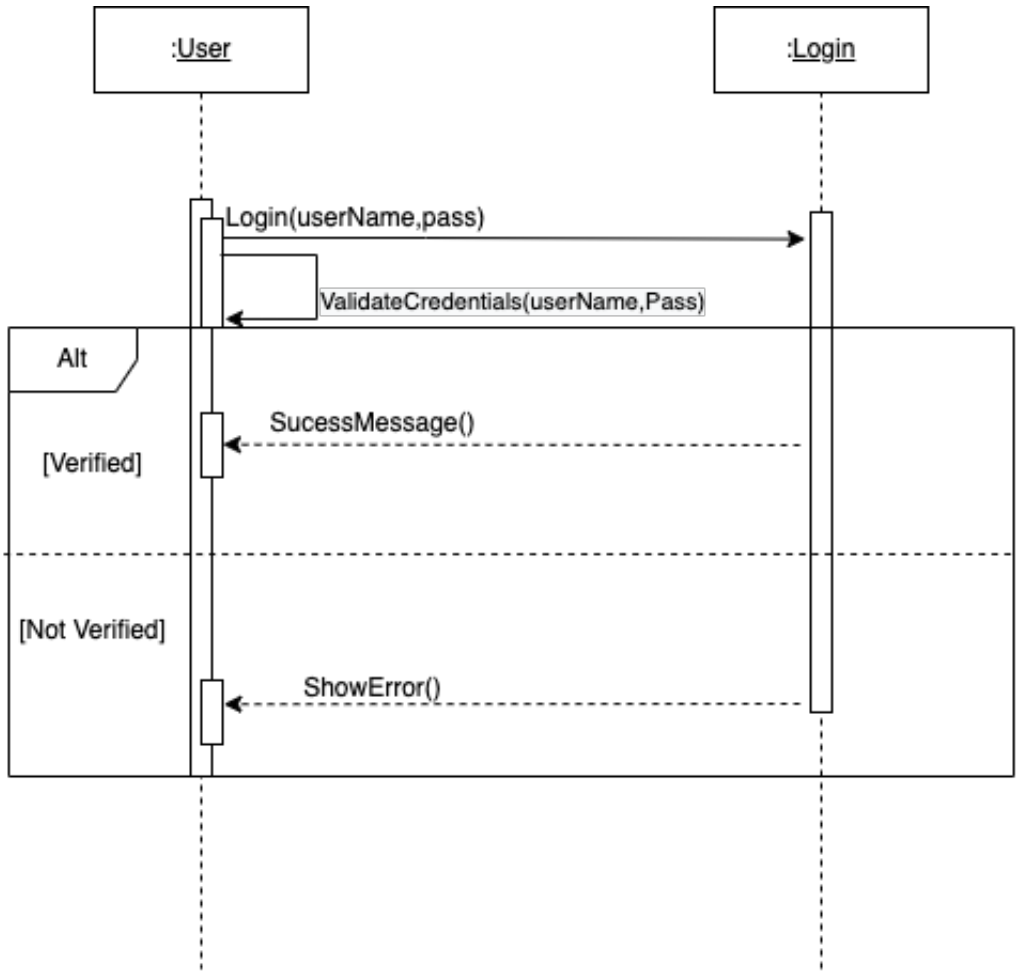

Figure 11: SD-02 Person Login

10.3 Take XML Format

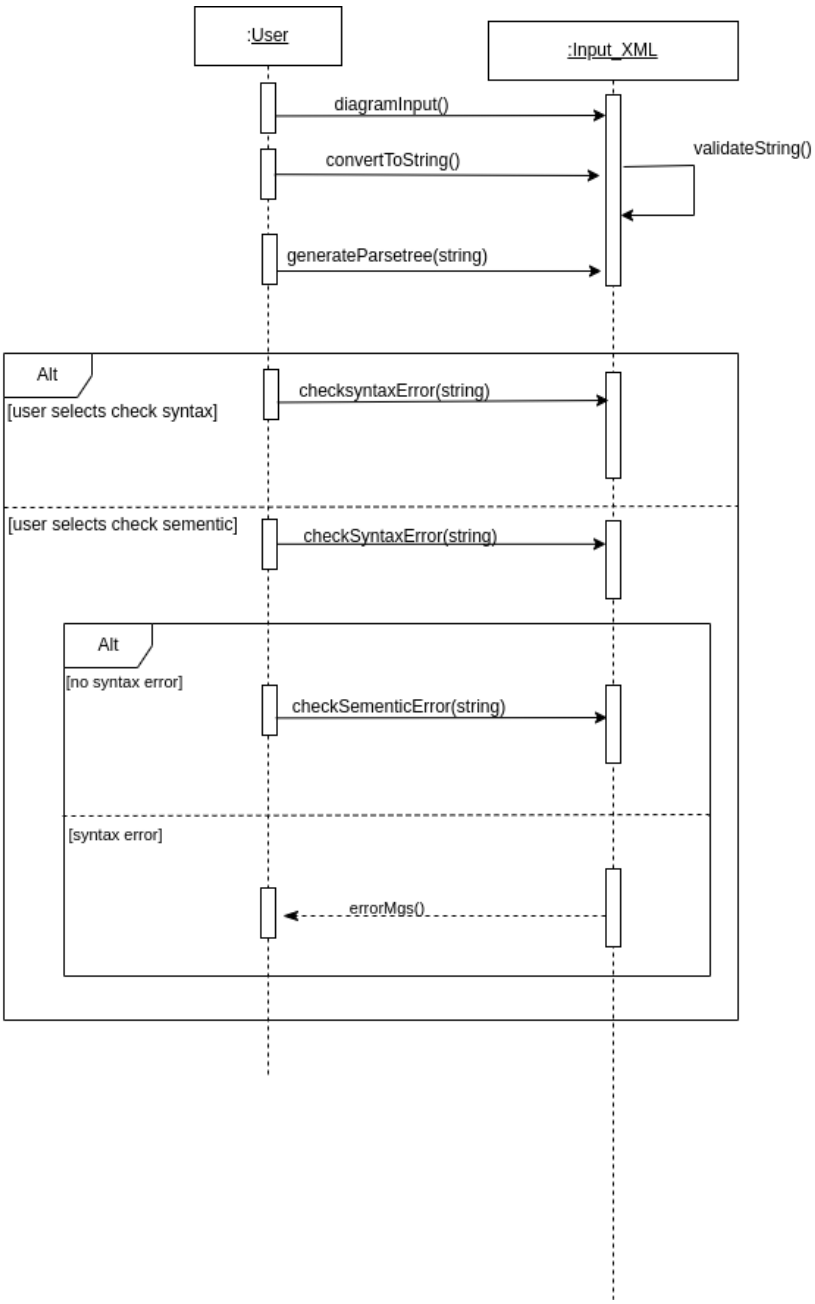

Figure 12: SD-03 Take XML Format

10.4 Release New Sentence

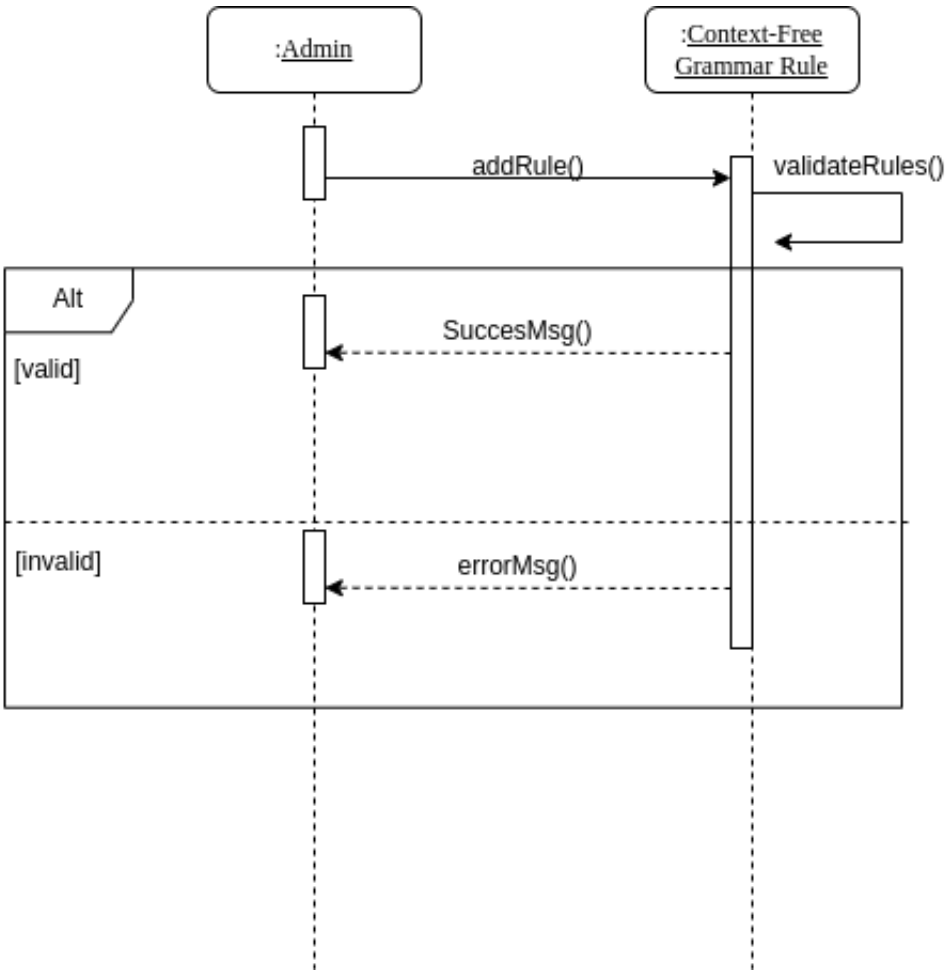

Figure 13: SD-04 Release New Sentence

11 Entity Relation Diagram

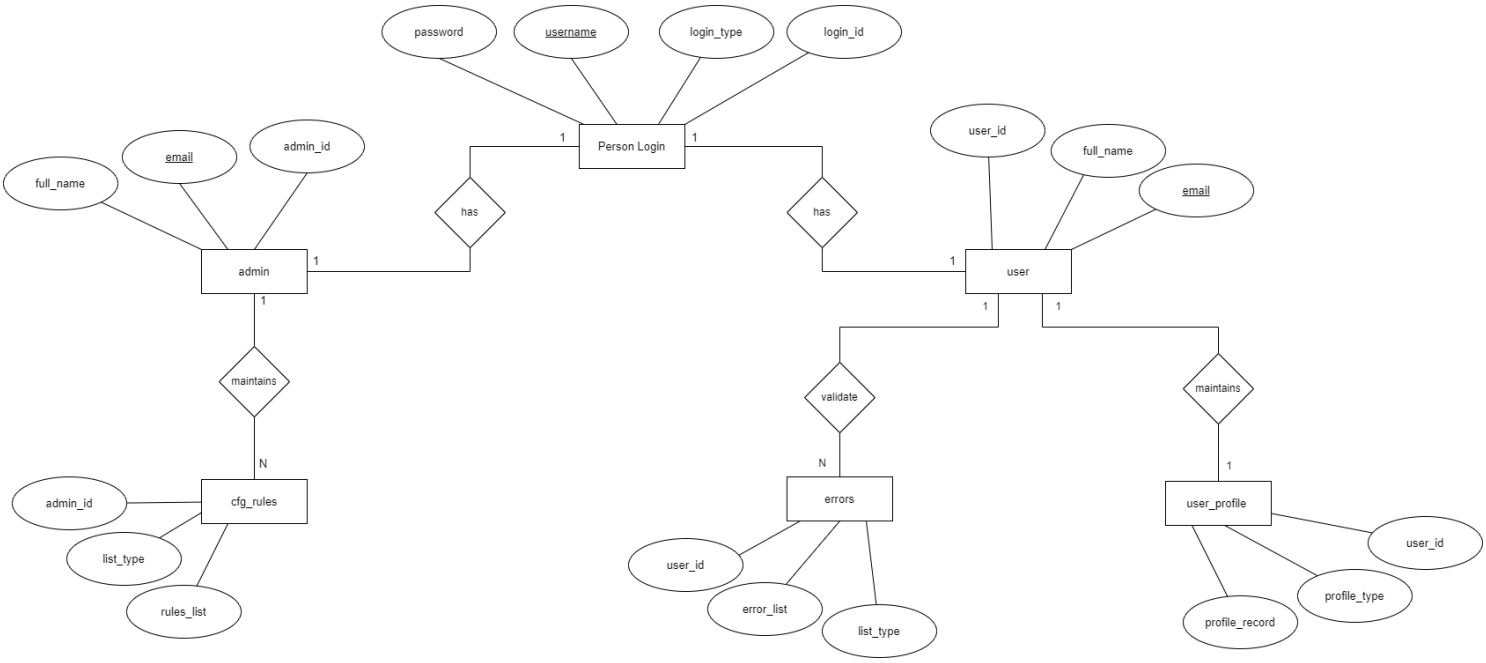

Figure 14: Entity Relation Diagram

12 Architecture Diagram

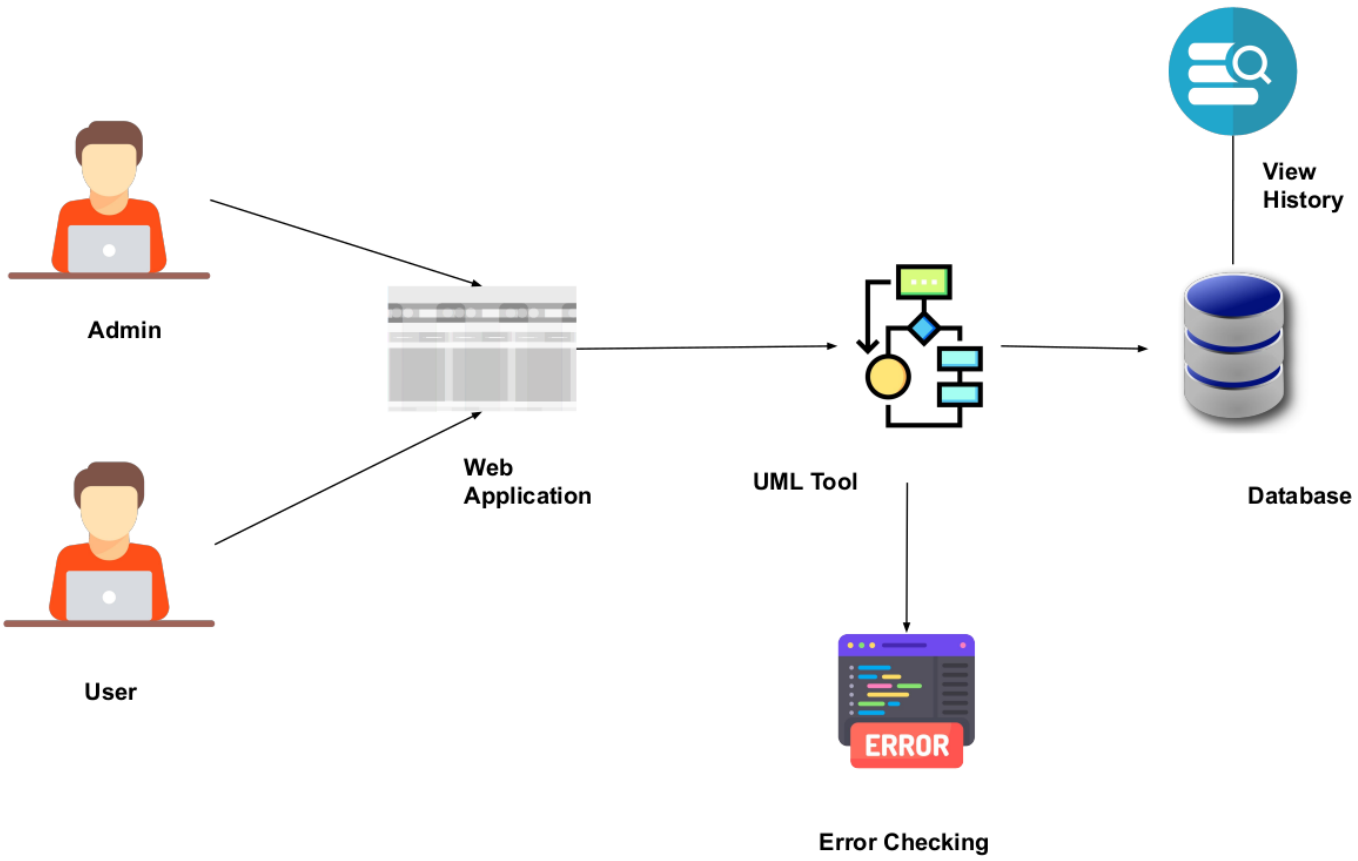

Figure 15: Architecture Diagram

13 Test Cases

13.1 Login

Table 17: TC-01 Login

Test case ID	**Input Values**	**Expected Output**	**Actual Output**	**Pass/Fail Status**
Valid Username and password	email=xyz.@gmail.com, password=abc@	Login Successful	Login Successful	Passed
Invalid Username	email=123@gmail.com	nvalide Email format	Invalid Email format	Passed
Invalid password	password=¿/dkaxasd	Password does not match.Try again!	The password does not match.Try again!	Passed
Empty username and password field	email= ,password=	Fill the fields	Fill the fields	Passed
Password with incorrect case	email=123@gmail.com, password=./dkasda	Invalid Credentials	Invalid Credentials	Passed

13.2 Database Connection

Table 18: TC-02 Database Connection

Test case ID	**Input Values**	**Expected Output**	**Actual Output**	**Pass/Fail Status**
Database connection	Var connection= mysql.createConnection (vars)	SQL CON-NECT SUC-CESS-FUL	SQL CON-NECT SUC-CESS-FUL	Passed
Database Validation	Var interval =set-Interval(function() connection.ping()	Ping suc-cessful	Ping suc-cessful	Passed
Database Security checks	Exceptional Han-dling	Securely transfer-ring the data	No out-put	Does not exist

13.3 Node js

Table 19: TC-03 Node js

Test case ID	**Input Values**	**Expected Output**	**Actual Output**	**Pass/Fail Status**
Functionality		All work-ing fine	Working	Passed
Error handling	If(err)=console.log() else console.log()	Errors are han-dled execute	Errors are han-dled	Passed
Asynchronous	loginUser=async()	Await,promises execute	Executing in nor-mal flow	Passed

13.4 Express Route

Table 20: TC-04 Node Express Route

Test case ID	Input Values	Expected Output	Actual Output	Pass/Fail Status
Successful Request	router.post("/login")	Requesting works	Routing	Passed
Invalid Request	router .post("/directory")	Some Error occur	Some Error occur	Passed
Error Handling	if(err)console.log(err)	Handling errors	Some Error occur	Passed
Authentication	sql2=($(email),$(pass)	Some error occur in checking email	Some error occur in checking email	Passed

13.5 XML Parsing

Table 21: TC-05 XML Parsing

Test case ID	Input Values	Expected Output	Actual Output	Pass/Fail Status
Valid XML	ReadFile(xml)	Handle file read error	File is valide	Passed
inflate XML	parser.pareseString(data)	Handling parsing error if any	retuning	Passed
XML Child and Parent	if(mxcellarray[i]=1)	It means its a class	It means its a class	Passed
XML schema validation	Into Class and attribute and methods	Differnetiate the classes	Differnetiate the class	Passed

14 References

- https://www.uml-diagrams.org/uml-25-diagrams.html
- https://nulab.com/learn/software-development/intro-uml-diagram-types-templates/
- http://www.cs.sjsu.edu/~pearce/modules/lectures/uml2/index.htm
- https://link.springer.com/chapter/10.1007/978-3-642-11659-9_22