ProUML

Amer Yono: Major in SE
Corey Taylor: Major in SE
Marin Mirasol: Major in SE

Course Instructor: Simon Fan
Faculty Advisor: Simon Fan

Industry Sponsor: CSUSM CSTEM
Project Mentor: Simon Fan
Professor

A capstone project report submitted to the faculty of
Computer Science and Information Systems
California State University, San Marcos

March 2023
(Version 3.5)

Technical Report Series: CSU-SM-CSIS-Class2023-Sec-001-Team-001

Table of Contents

1. Abstract

With the complexity of software increasing, as technology becomes more advanced, many software engineers today need to use tools to make complex software systems more understandable. One tool that many software engineers use is Unified Modeling Language (UML), which is a modeling language that visualizes software designs from various perspectives. Although UML helps to make software design more understandable, it is very time-consuming because engineers need to manually construct UML diagrams, and then translate diagrams into cod.e, and vice versa. With ProUML, engineers can easily import source code from software projects to translate automatically into UML. Conversely, they can do the opposite, which is to translate constructed UML diagrams into source code. Not only is ProUML convenient for design activities, but it can also be easily expanded to support newer programming languages if other developers are willing to implement additional language support for our system. Overall, with ProUML, our goals are to make software development more efficient, less costly, and less time-consuming.

2. Report Revision History

2.1. Changes in Version 1.5

Several revisions have been made in this technical report version to fix issues with the last report version or include updates that have been made with our progress of developing ProUML. One change made in this version was the further clarification of the types of users we want to support in section 3.3. "Objectives," as we clarified the "variety of users" that we want to support includes developers, students, and instructors that use UML and may collaborate. Another change in this report version was section 4.1.2. "User Groups," as we made our user groups more specific by including different types of users based on the collaboration feature we plan to implement in ProUML. The user groups that we included are "ProUML Project Owners" and "ProUML Project Collaborators." As a result of expanding the user groups, we also edited the use case diagram in section 4.1.3.1. "Project Scope" to reflect the two user groups we specified in this version. Section 4.1.3.2. "User Scenarios" was also changed, as we added another user scenario to "Edit UML Diagram." This scenario was added since collaborators can edit ProUML projects that are already created from the project owner's user scenario to "Create UML Diagram." Additional changes in the report's format and wording have also been made to make the report more concise and easier to read, which include the rewording of some requirements to make the descriptions shorter and more concise, and the additions of captions below figures and tables for references and descriptions. The bottom margins have also been increased to make more room for page numbers and our team number has been added on the top right of each page. After making these changes, the report is easier to read and has been updated to reflect the latest progress of ProUML's development. In the future, we will continue to make revisions to this report to reflect any changes we make and to improve the readability of this report.

2.2. Changes in Version 2.0

Several additions have been made in this technical report version as we have progressed with our development of ProUML. As we have implemented some of our production code, we have designed some behavior diagrams. The behavior diagrams, which can be found in the added Section 6.4 “Behavioral Design,” includes two sequence diagrams, which visualize the sequences of how ProUML works in different scenarios. Additionally, since we are practicing Test Driven Development (TDD), we have also added Section 8 “System Testing,” which includes several sub-sections that discuss how we set up our testing environment, ran our test cases, designed our test suite and cases, and summaries of our execution report. As a result of adding our testing processes, we have also added Appendix T and Appendix TE, which go more in-depth into our tests and test executions. As we continue developing ProUML, we will continue to add more sections and will make revisions in future report versions to reflect our latest progress.

2.3. Changes in Version 2.5

A couple of minor tweaks have been made in this technical report version, which include wording changes, diagram tweaks, and a diagram addition. We made some wording changes to make this report more concise. In terms of the diagram tweaks, we initially made some wording mistakes on some of our diagrams in version 2.0, which include our high level architectural diagram in section 6.1 “Architectural Design,” and our general user activity diagram in Section 6.2 “Behavioral Design.” In addition to tweaking our existing diagrams, we also added an additional transpiler activity diagram in Section 6.2 “Behavioral Design” to expand on the “Transpile Program into JSON” activity on the general user activity diagram. With these changes, this report should be more concise syntactically and our diagrams in this report version should be easier to understand.

2.4. Changes in Version 3.0

Some additions have been made in this report to reflect the progress of implementing the frontend and having it function with the backend of ProUML. First, we included the additional tools that we began using during this phase of development to Section 7.1 “Programming Languages and Tools.” The added tools in this section include: Next.js used for implementing our frontend in React with server-side rendering to function with our backend, Redis used for the live sharing feature, Postgres for database storage, and Tailwind for customizing our user interface. Additionally, as a result of implementing our frontend, we added another activity diagram to Section 6.4 “Behavior Design” to demonstrate some of the user interaction activities. Specifically, this diagram addition portrays the user activities when in the diagram editor page on ProUML, which is a primary feature of our application. After including these revisions, the diagram editor logic, the frontend of ProUML, and how the frontend works with the backend should be more clear.

2.5. Changes in Version 3.5

One minor tweak has been made in this technical report version, as we added Section 6.5 “Design Alternatives & Decision Rationale” and Section 7.4 “Implementation Alternatives & Decision Rationale.” The purpose of Section 6.5 is to visualize the design of the live-sharing feature in a diagram format. This diagram shows the communications between users, servers, and databases. Section 7.4 serves to go more in-depth about our live-sharing design and explains Section 6.5’s diagram in more detail. With the changes made in this report version, our live-sharing feature works with communications between users, servers, and databases.

3. Problem Statement

3.1. Business Background

ProUML is an existing web application that allows users to translate from Java to UML. The current system of ProUML has been developed using a variety of languages and frameworks. Some of the languages used to develop ProUML include Go for hosting the website and React in TypeScript for the front end of the application. The frameworks that are used in ProUML include AntV for frontend diagram design support and supabase for user authentication and account management. ProUML is also open source, so other developers can also contribute to the application’s development. Although ProUML is already an existing application, it is still incomplete and needs more functionalities and feature support to be a fully-functioning web application that is beneficial for software engineers and students.

3.2. Needs

There are many reasons why ProUML is needed in today’s software engineering environment. For one, ProUML helps to make the process of software development more efficient, as it takes away the time required to translate between Java to UML and vice versa. Additionally, ProUML promotes good software engineering practices because using UML helps software engineers easily visualize large and complex software systems, which can improve the overall understanding of more complex software systems. Lastly, with the open source development of ProUML, the capabilities are endless with this web application, as we and other developers can collaborate to make a system that is more capable, supports a variety of programming languages, and that provides many different tools that can be used by software engineers and students for designing software. Overall, ProUML is an application that can be beneficial to the software engineering industry and its development is necessary since it can help to improve the efficiency of software development, can improve the understanding of software, and can easily be expanded to an even greater and more useful system.

3.3. Objectives

We hope to extend the current system of ProUML into a system that can support a variety of users, such as developers, students, and instructors that utilize UML and may collaborate. We also hope that our system can support a variety of useful features that can make the application more usable and beneficial for use. For instance, a feature that we plan to implement, which would be a very helpful feature for users, is to give the users the ability to collaborate with others on ProUML diagrams. From working on ProUML, we also want to learn more about frontend web frameworks, such as AntV. As a whole, our team has many goals that we plan to achieve while working on ProUML, and these goals include expanding ProUML's current system, making ProUML more usable, and using as well as learning more about the frameworks used in ProUML.

4. Requirements

4.1. User Requirements

4.1.1. Glossary of Relevant Domain Terminology

Unified Modeling Language (UML): Language used to visualize software code into classes with their associations to other classes in the code.

UML Class Diagram: UML visualization of the code in an application, package or file in an application.

Java: Object-oriented programming language that can be used to develop many different types of software applications.

Java Keywords: Important words in Java that define certain aspects of the code.

Variables: Pieces of data in Java that can store specific values based on its data type.

Methods: Functions that can repeatedly be used when their implementations are needed.

Class: Keyword in Java that defines special object(s) that have specific properties, which can include variables and/or methods.

Class Attributes: The different variables that are defined and used within a class.

Class Functions: The methods that are defined and used within a class.

Class Associations: How classes interact and relate with each other.

Object: An instance of a class.

Parameters: Variables or data that are passed to methods when used.

Scope: The region, within code, where objects or variables can be used.

Interface: Java keyword that indicates when a class of empty methods can inherit child classes of shared properties.

Implements: Java keyword that indicates when a class is a child of an interface class and implements the empty methods from the interface class.

Abstract: Java keyword that indicates when a class of implemented methods can inherit child classes of shared properties.

Extends: Java keyword that indicates when a class is a child of an abstract class, which may implement additional methods or override the implemented methods of the abstract class.

Static: Java keyword that indicates when an object or variable defined in a class is specific to the entire class, rather than different objects of the class.

Public: Java keyword that indicates when an object or variable defined in a class can be accessed outside of the class's implementation.

Private: Java keyword that indicates when an object or variable defined in a class can only be accessed within the class's implementation.

Final: Java keyword that indicates when the properties or values of a class or variable cannot change after being declared.

Data Type: Defines what type of data can be stored in a variable, and comes in the Java keywords byte, short, int, long, float, double, boolean, and char.

Byte: Java keyword that defines a data type of a variable that can store whole numbers from -128 to 127.

Short: Java keyword that defines the data type of a variable that stores whole numbers from -32,768 to 32,767.

Int: Java keyword that defines the data type of a variable that stores whole numbers from -2,147,483,648 to 2,147,483,647.

Long: Java keyword that defines the data type of a variable that stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.

Float: Java keyword that defines the data type of a variable that stores fractional numbers with up to 6 to 7 decimal points.

Double: Java keyword that defines the data type of a variable that stores fractional numbers with up to 15 decimal points.

Boolean: Java keyword that defines the data type of a variable that stores true or false values.

Char: Java keyword that defines the data type of a variable that stores a single character.

JavaScript Object Notation (JSON): File format that stores data structures and objects.

Parse: Analyzing and organizing text into parts.

Java Package: A folder of several Java files.

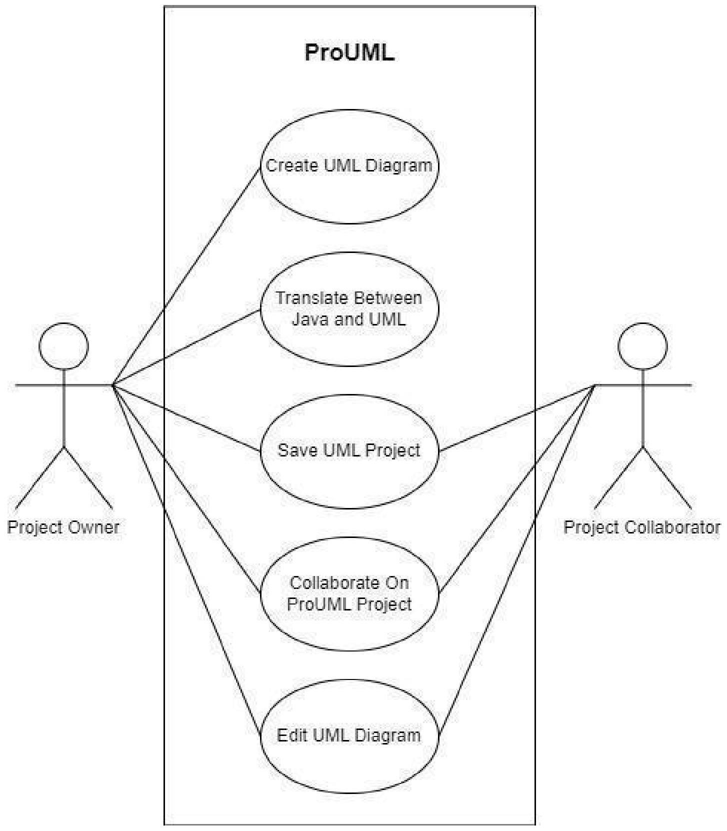
4.1.2. User Groups

ProUML Project Owners: Want to use the application to create UML diagrams, translate from Java code to a UML class diagram, export a UML class diagram to Java boilerplate code, save UML projects, and collaborate on UML projects with others. These users have ultimate ownership over their created diagram projects, as they can edit, delete, and share their projects with possible collaborators.

ProUML Project Collaborators: Want to collaborate with a ProUML project owner to edit and contribute to the project owner's project. In order to have access to a project owner's project and collaborate on it, the project owner must explicitly share their project with these types of users. These users have partial ownership over projects they collaborate in, as they cannot delete an owner's project.

4.1.3. Functional Requirements

4.1.3.1. Project Scope:



Refer to **Table 4.1** in **Appendix U** for more information

4.1.3.2. User Scenarios

UC-001 ("Create UML Diagram"): A primary-level use case, where the user is allowed to create a diagram that is either pre-configured and structured from a template or is empty for the user to manipulate via a diagram editor. (Refer to **Table 4.2** in **Appendix U**)

UC-002 ("Translate Between Java and UML"): A primary-level use case, where the user is able to import Java files and have them translated into UML diagram form, while also being able to export completed UML diagrams into Java source code.
(Refer to **Table 4.3** in **Appendix U**)

UC-003 ("Save UML Project"): A primary-level use case, where the user can save any progress of a UML diagram and create/resume any other diagram in the editor.
(Refer to **Table 4.4** in **Appendix U**)

UC-004 ("Collaborate On ProUML Project"): A primary-level use case, which allows the user to collaborate on a project with other users that are actively working on the same project, with real-time updates. (Refer to **Table 4.5** in **Appendix U**)

UC-005 ("Edit UML Diagram"): A primary-level use case, where the user is allowed to edit an already-existing diagram that either appears in their saved diagrams or in their collaborated projects. (Refer to **Table 4.6** in **Appendix U**)

4.1.3.3. User Functional Requirements:

UF-A: Users shall be able to create and edit UML diagrams.
(Refer to **Table 4.7** in **Appendix R**)

UF-B: Users shall be able to translate their Java code to UML and vice versa.
(Refer to **Table 4.8** in **Appendix R**)

UF-C: Users shall be able to save their UML diagram data on ProUML.
(Refer to **Table 4.9** in **Appendix R**)

UF-D: Users shall be able to collaborate with others on ProUML projects.
(Refer to **Table 4.10** in **Appendix R**)

4.1.4. Non-functional Requirements

4.1.4.1. Product: Usability Requirements

UP-01: Users shall be able to use a web application that is extendable, customizable, and easy to use. (Refer to **Table 4.11** in **Appendix R**)

4.2. System Requirements

4.2.1. Functional Requirements

4.2.1.1. System Functional Requirements

SF-A-01: The system shall provide a diagram editor designed specifically for UML diagram editing. (Refer to **Table 4.12** in **Appendix R**)

SF-A-02: The system shall provide a button that allows users to add UML class shapes to their UML diagrams. (Refer to **Table 4.13** in **Appendix R**)

SF-A-03: The system shall provide a sidebar for UML class shapes when these shapes are clicked on, where the class name, class associations, whether the class is an interface or not, attributes and methods can be edited by the users. (Refer to **Table 4.14** in **Appendix R**)

SF-B-01: The system shall be able to comprehend Java code into classes, class associations, and class functions by understanding Java class names, attributes, and keywords for Java to UML translation. (Refer to **Table 4.15** in **Appendix R**)

SF-B-02: The system shall be able to organize completed UML diagrams into Java code for UML to Java translation. (Refer to **Table 4.16** in **Appendix R**)

SF-B-03: The system shall preserve imported code that is not needed in UML diagram translation for UML to Java translation. (Refer to **Table 4.17** in **Appendix R**)

SF-C-01: The system shall support user profiles with authentication.
(Refer to **Table 4.18** in **Appendix R**)

SF-C-02: The system shall save user data and associate the data with specific user profiles in a database. (Refer to **Table 4.19** in **Appendix R**)

SF-C-03: The system shall translate UML diagrams to a text file that can be saved in a database. (Refer to **Table 4.20** in **Appendix R**)

SF-D-01: The system shall support live sharing by using user-to-user communication. (Refer to **Table 4.21** in **Appendix R**)

SF-D-02: The system shall be able to save collaborated projects in real time. (Refer to **Table 4.22** in **Appendix R**)

4.2.2. Non-functional Requirements

4.2.2.1. Product: Usability Requirements

SP-01-01: The system shall provide a user login page where users can log into their accounts. (Refer to **Table 4.23** in **Appendix R**)

SP-01-02: The system shall provide a dashboard page with the ability to import source code to translate to Java code, start a UML diagram from scratch, or choose a template design pattern diagram to start from. (Refer to **Table 4.24** in **Appendix R**)

SP-01-03: The system shall allow users to edit the user interface with dark and light modes. (Refer to **Table 4.25** in **Appendix R**)

4.3. Requirements Trace Table

User Requirements	**System Requirements**
Users shall be able to create and edit UML diagrams.	The system shall provide a diagram editor designed specifically for UML diagram editing.
	The system shall provide a button that allows users to add UML class shapes to their UML diagrams.
	The system shall provide a sidebar for UML class shapes when these shapes are clicked on, where the class name, class associations, whether the class is an interface or not, and attributes and methods can be edited by the users.
Users shall be able to translate their Java code to UML and vice versa.	The system shall be able to comprehend Java code into classes, class associations, and class functions by understanding Java class names, attributes, and keywords.
	The system shall be able to organize completed UML diagrams into Java code.
	The system shall preserve imported code that is not needed in UML diagram translation.

Users shall be able to save their UML diagram data on ProUML.	The system shall support user profiles with authentication.
	The system shall save user data and associate the data with specific user profiles in a database.
	The system shall translate UML diagrams to a text file that can be saved in a database.
Users shall be able to collaborate with others on ProUML projects.	The system shall support live sharing by using user-to-user communication.
	The system shall be able to save collaborative projects in real-time.
Users shall be able to use a web application that is extendable, customizable, and easy to use.	The system shall provide a user login page where users can log into their accounts.
	The system shall provide a dashboard page with the ability to import source code to translate to Java code, start a UML diagram from scratch, or choose a template design pattern diagram to start from.
	The system shall allow users to edit the user interface with dark and light modes.

Refer to **Table 4.26** in **Appendix R** for more information

5. Exploratory Studies

5.1. Relevant Development Frameworks

Frameworks that we wanted to use in our system are React, Fiber, Material-UI, and AntV. React is a frontend JavaScript framework that allows developers to make applications that are declarative, efficient, and flexible [11]. There are dozens of other JavaScript frameworks that allow for the same modularity as React, but we found React to be the most beneficial with its ease of use in state management [11]. We are easily able to wrap our application around any context we want, such as an authentication context, which would allow us to pass down data to child components without having to pass down props through each child component manually [11]. This is especially useful when creating deeply nested components that require the user's authorization session state [11]. Although we found that React is greatest in the state management aspect, we did find that its routing system had major flaws, especially when it came to search engine optimization (SEO), which would be a huge requirement for us in the future when we want to advertise our service to developers in the industry [11]. There are other frontend frameworks that alleviate the pains of SEO, but we decided to stick with React because of how we are integrating it with Go and Fiber.

This brings us to why we decided to use Fiber as our HTTP handler framework with Go. Fiber allows us to integrate custom-made SEO tags per page such as the page's title and description inside a single HTML file [14]. This will make our React setup slightly similar to a NextJS setup, but not nearly the same [9]. NextJS automatically generates all HTML files to be served as static on runtime, but in our setup, we would dynamically generate these tags on every load [9]. This could be seen as repetitive and unnecessary, but this allows us to also pass along authorization sessions to the user when they first fetch our website which would allow us to save an extra HTTP request on the client's side [9].

For the frontend user interface styling, we decided to use Material-UI compared to others such as Bootstrap [11]. This is because of the easily customizable vast amount of features, but simultaneously complex functionality Material-UI offers [11]. It holds a comprehensive suite of UI tools that will help us ship our product out as fast as possible, while also looking stunning to the human eye [11]. For our frontend class diagram editor, we decided to use a framework to help us navigate the HTML canvas. It would make it considerably less challenging bearing in mind that shapes would need to be dragged onto the canvas, editable, and deletable. Our framework of choice is AntV X6 [4]. We decided to use AntV because of its high level of usability and customization [3]. It provides us with numerous interactive components, custom node capabilities, and on-demand listeners to facilitate the construction of the UML class diagrams. AntV enables us to empower developers with real-time collaborative editing, through the use of its graph listeners, alongside Fiber, connecting all online document viewers through WebSockets [3].

5.2. Relevant Solution Techniques

Some relevant solution techniques that are open-source include AntV XFlow, PlantUML, and ownCloud. AntV XFlow uses AntV X6, our diagramming framework, to create a complete diagramming application [2]. We could not use XFlow as part of our solution for ProUML because of its lack of enablement in customization, as we want to design our own diagram editor with UML class-specific capabilities [5]. We found it very difficult to integrate custom React components into the diagram, custom edges with text on multiple sides, and in-depth functionalities with XFlow, but we are able to use it as a starting point to see how they integrated their own tools into the application. XFlow has many features built on top of X6 that allow for a complete diagramming solution such as grouping shapes, adding, removing, selecting, highlighting, and moving shapes, similar to draw.io, but their allowance of an open-source product does not save us completely [3]. We are able to look over the open-source code, to help implement some of the functionalities they did, but little to none of their documentation is written in English. This is going to be the most difficult roadblock we are going to encounter when developing the ProUML application.

For the diagram layout generator, we will be using PlantUML's open-source code to assist us in the development of our own diagrams based on the user's imported code PlantUML also has an algorithm for creating UML diagrams, which would also help us with generating the layout of our UML diagrams after we transpile Java source code into UML diagram form [1].

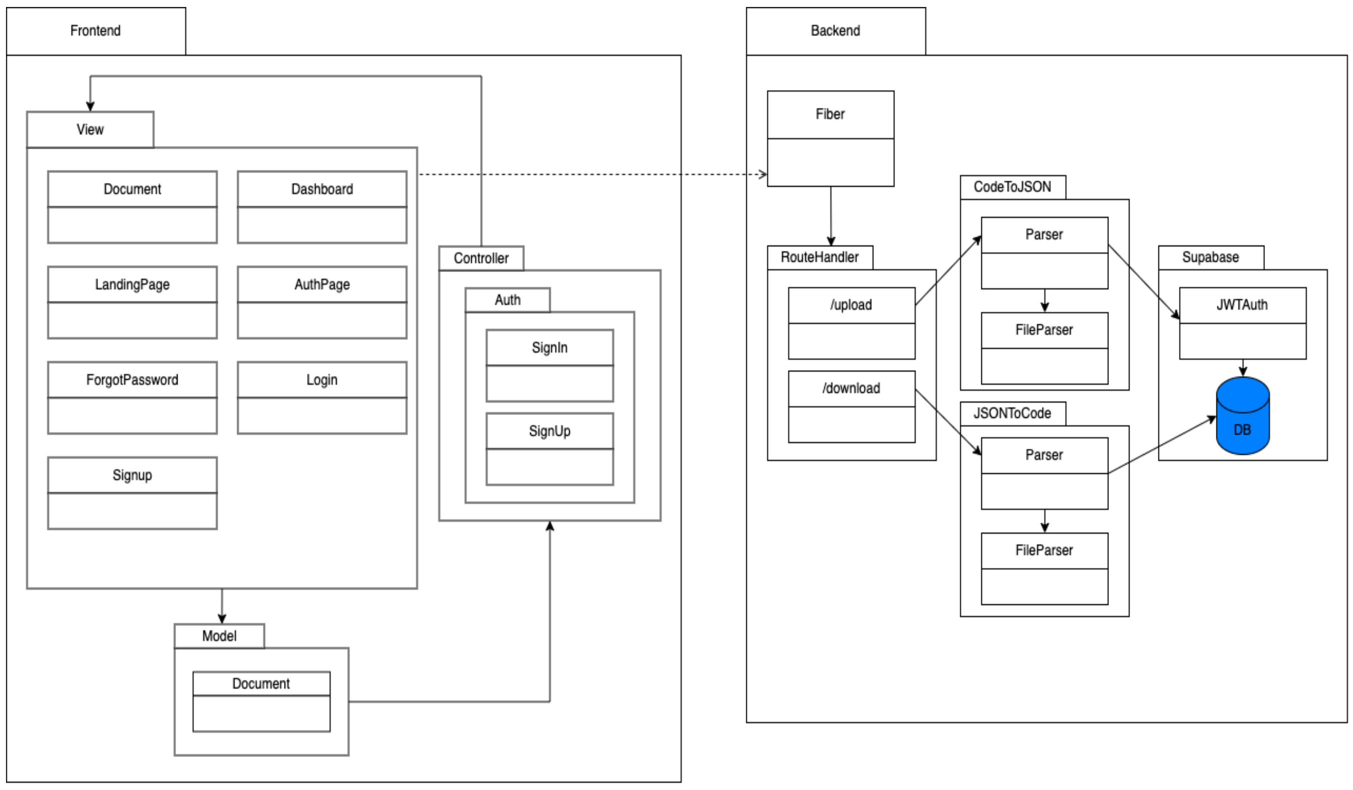
Another open-source tool we could use is ownCloud, a Google Drive-like library that enables users to share and collaborate with each other on documents [6]. We consider this resource because we want our application to have an environment where users can collaborate with each other on ProUML projects in real time. After researching these open-source libraries and tools, we make sure to utilize them to the best of our abilities in the development of ProUML.

5.3. Broader Impacts

With ProUML, we hope to make a difference in software development by making the process of development simpler and more efficient for aspiring software engineers, students, or anyone who wants to develop software. Our system can change modern practices when it comes to translating UML and code by allowing engineers to allocate more time to either UML modeling or coding implementation, which can improve industry standards. As stated in our project abstract, software complexity is constantly increasing over time, and with ProUML, we can help reduce the confusion in complex software by making it simpler to construct UML diagrams to help visualize these complex software systems.

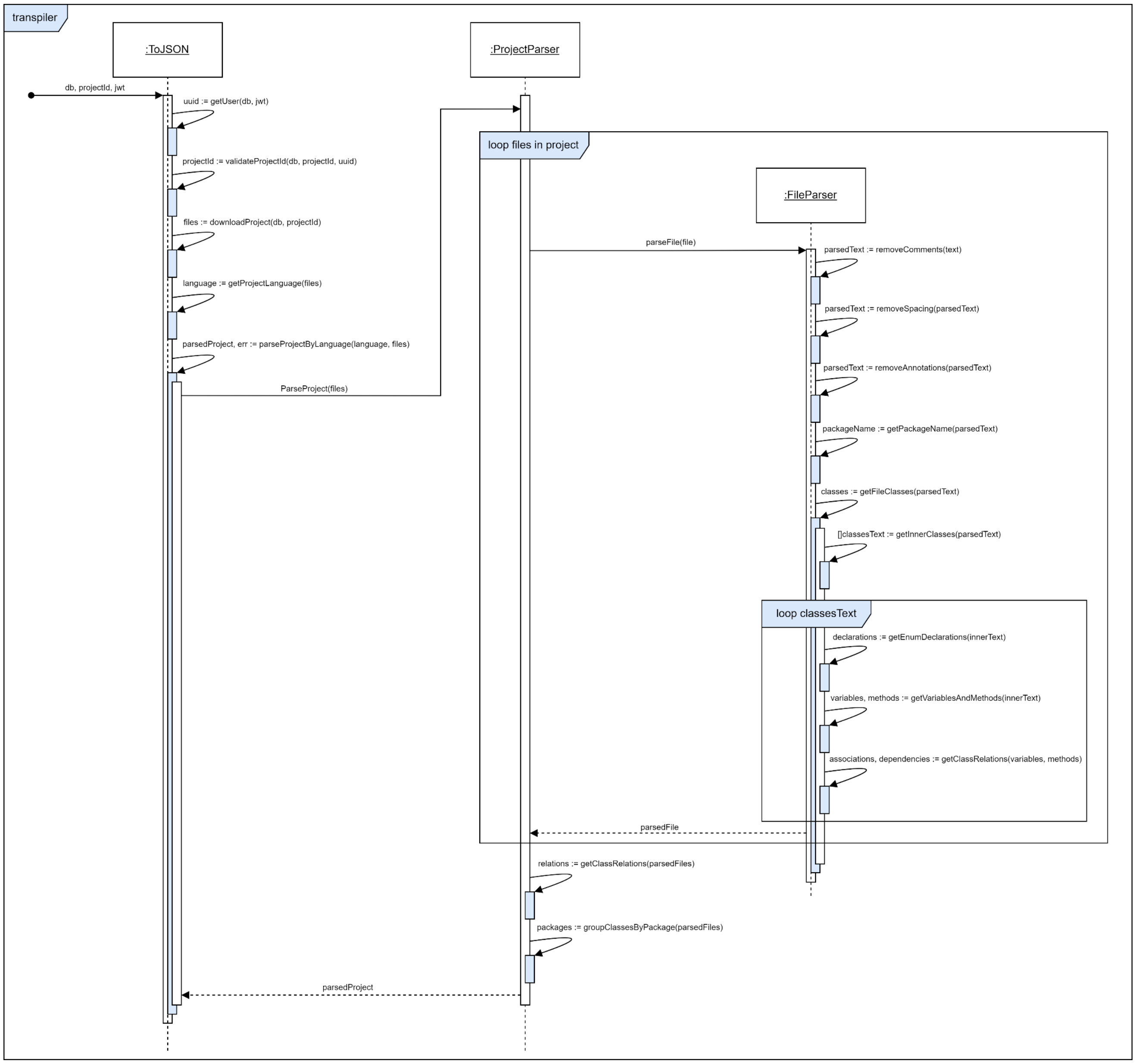
6. System Design

6.1. Architectural Design

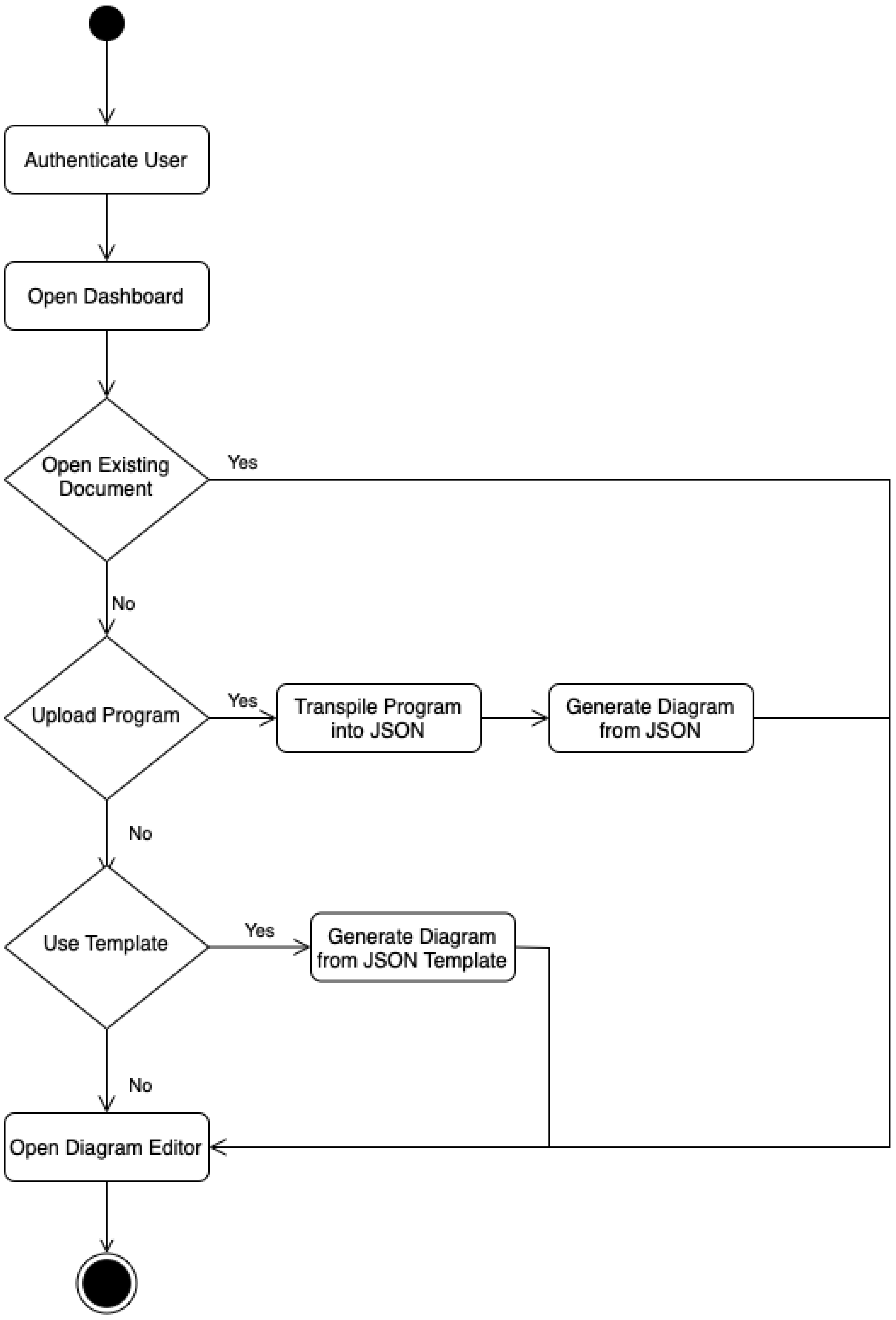


This figure visualizes the high-level architectural design of ProUML

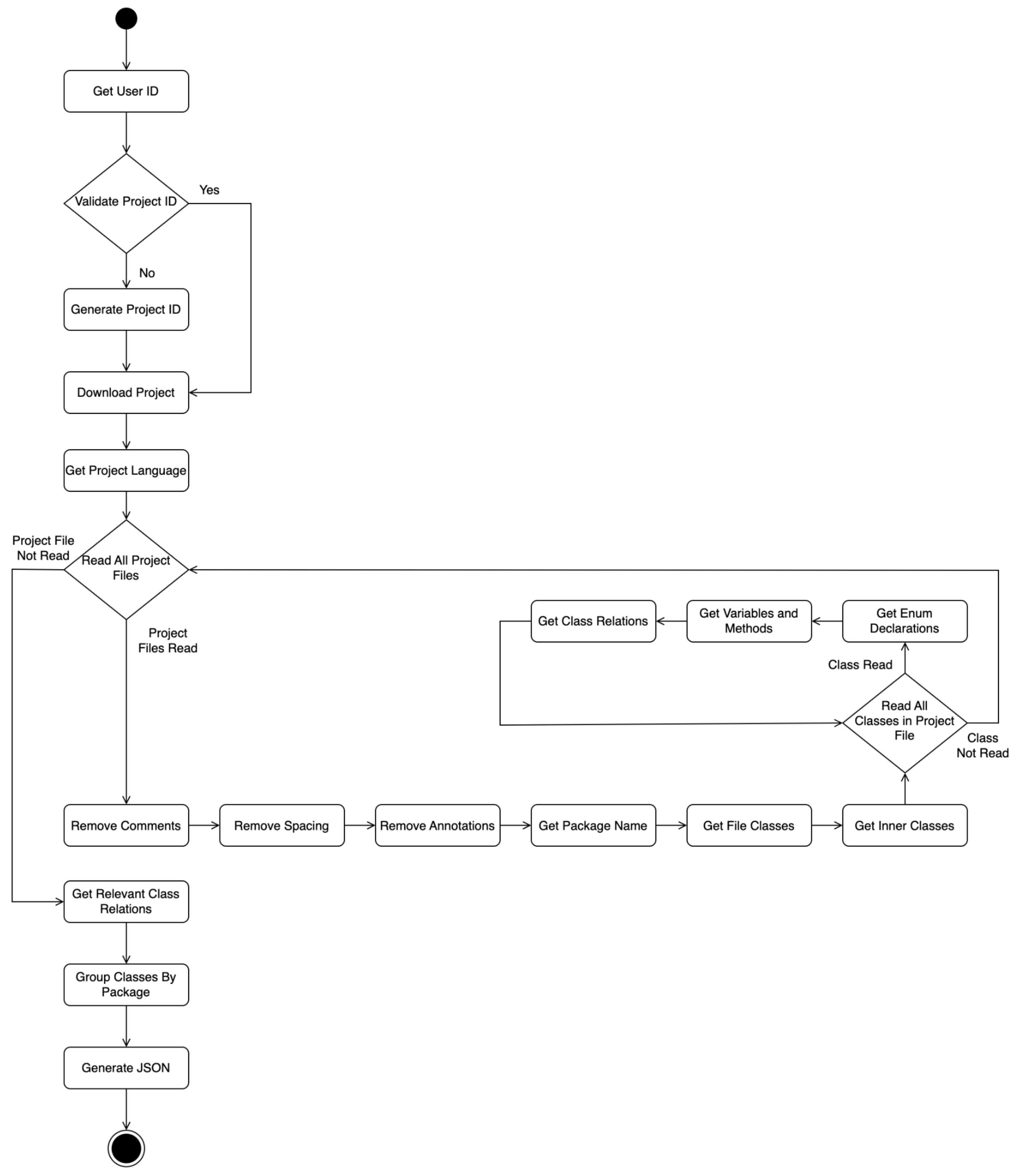
6.4. Behavioral Design


transpiler
:ToJSON
:ProjectParser
db, projectId, jwt
uuid := getUser(db, jwt)
projectId := validateProjectId(db, projectId, uuid)
files := downloadProject(db, projectId)
language := getProjectLanguage(files)
parsedProject, err := parseProjectByLanguage(language, files)
ParseProject(files)
loop files in project
:FileParser
parseFile(file)
parsedText := removeComments(text)
parsedText := removeSpacing(parsedText)
parsedText := removeAnnotations(parsedText)
packageName := getPackageName(parsedText)
classes := getFileClasses(parsedText)
[]classesText := getInnerClasses(parsedText)
loop classesText
declarations := getEnumDeclarations(innerText)
variables, methods := getVariablesAndMethods(innerText)
associations, dependencies := getClassRelations(variables, methods)
parsedFile
relations := getClassRelations(parsedFiles)
packages := groupClassesByPackage(parsedFiles)
parsedProject

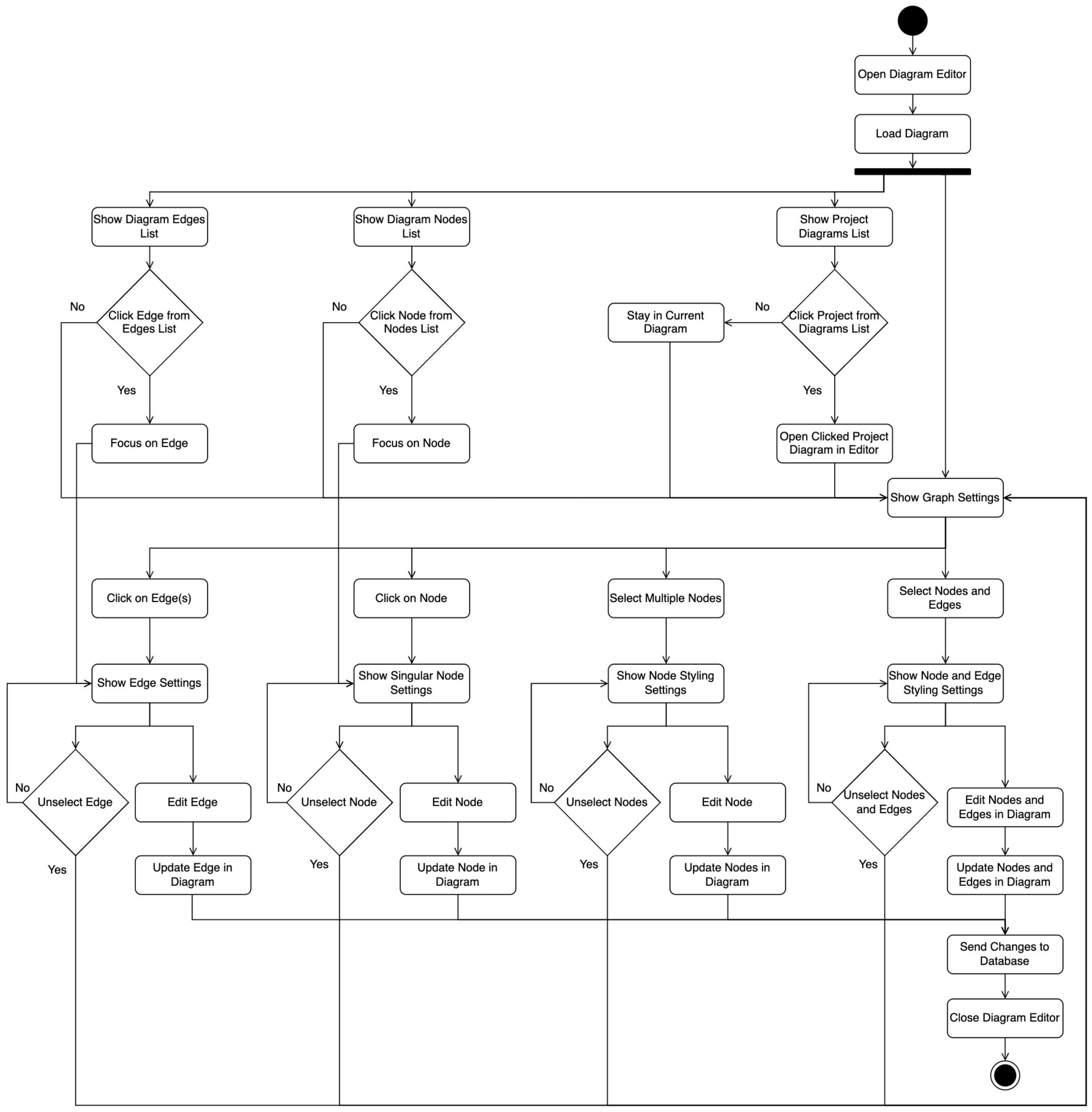
This figure visualizes the functional sequence of the transpiler in ProUML



This figure visualizes the general user activities of opening a diagram on ProUML

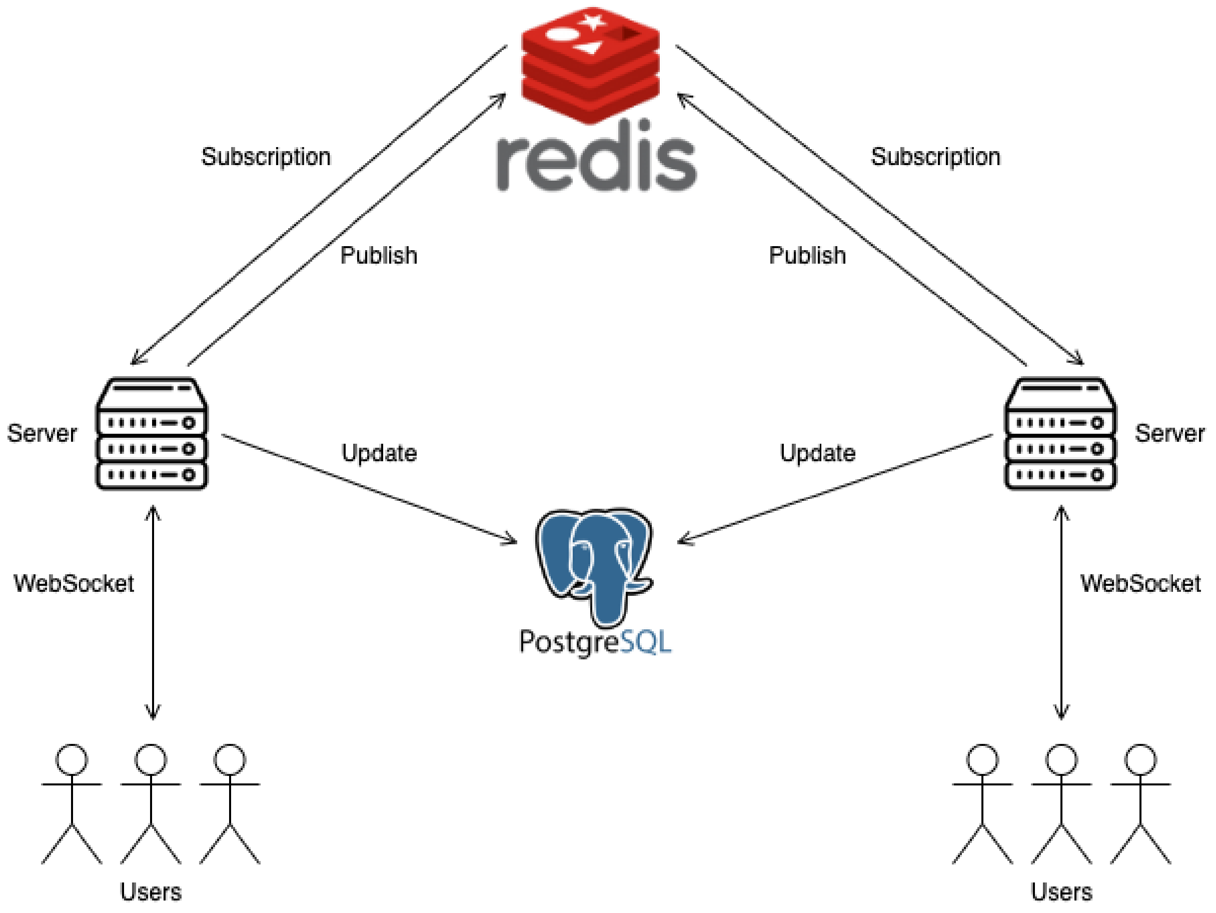


This figure visualizes the activities of the transpiler in ProUML



This figure visualizes the user activities of the diagram editor page in ProUML

6.5. Design Alternatives & Decision Rationale



This figure visualizes the live-sharing architectural design. (Refer to **Section 7.4)**

7. System Implementation

7.1. Programming Languages and Tools

Go: One of the languages used on the server-side, which handles server requests [2].

JavaScript/TypeScript: Another language used on the server-side, which handles client requests to the server to translate imported Java code to UML. It is also used on the client-side as well for the user interface functionalities [9][10].

React.js: Frontend framework that is used to design the user interface [11].

Supabase: Database library that allows SQL-based storage with user-authentication capabilities [12].

AntV: Diagram library, which enables the development of a custom diagram editor [3].

VSCode: An integrated development environment (IDE) that comes with several extensions and tools, which we will use to develop our application [16].

Next.js: Web development framework created by Vercel enabling us to create a React-based web application with server-side rendering [18].

Redis: Millisecond response time database that supports live user collaboration while editing diagrams [19].

Postgres: Relational database management system that will store user account information such as personal and diagram data [20].

Tailwind: A utility framework that has a variety of customizable CSS classes that can be used in React applications, which we use to design our user interface [21].

7.2. Coding Conventions

UpperCamelCase: Naming convention for declaring classes, interfaces, types, enums, decorators, and type parameters [7].

lowerCamelCase: Naming convention for declaring variables, parameters, functions, methods, properties, and modules aliases [7].

CONSTANT_CASE: Naming convention for declaring global constant values, including enum values [7].

Module Imports: File imports are lowerCamelCase while files are snake_case [7].

Variables Declarations: Always use const or let to declare variables. Use const by default, unless a variable needs to be reassigned. Never use var [7].

Exceptions: Always use new Error() when instantiating exceptions, instead of just calling Error(). Both forms create a new Error instance, but using new is more consistent with how other objects are instantiated [7].

Switch Statements: All switch statements must contain a default statement group, even if it contains no code [7].

Equality Checks: Always use triple equals (===) and not equals (!==). The double equality operators cause error prone type coercions that are hard to understand and slower to implement for JavaScript Virtual Machines [7].

7.3. Code Version Control

Git: A distributed version control software, which manages the versioning of code repositories in GitHub [8].

GitHub: A cloud-based service that allows the creation of repositories with the ability to share these repositories with team members and use Git for version control [15].

7.4. Implementation Alternatives & Decision Rational

The design in section 6.5 shows how our scaled servers interact with each other to support live sharing. When users make changes from one server, their changes get published through Redis to a pub/sub channel. Other servers that are subscribed to the same channel can then receive these updates and apply the changes to the shared document. The choice of Redis pub/sub as the communication medium offers several advantages, as discussed below.

Implementation Alternatives:

Direct server-to-server communication: Instead of using Redis pub/sub, servers could directly communicate with each other using HTTP, WebSocket, or any other suitable protocol. However, this method would require more complex connection management, and scalability would be a significant concern.

Message queue systems (e.g., RabbitMQ, Apache Kafka): These systems are designed for high-throughput messaging and provide strong durability guarantees. However, they can be more challenging to set up and maintain compared to Redis.

WebRTC: WebRTC enables peer-to-peer communication between clients without the need for a server. While this method can help reduce server load, it may introduce complexities related to security and data consistency.

Decision Rationale:

Simplicity: Redis is easy to set up, configure, and maintain. It offers a straightforward way to implement a publish-subscribe pattern without the need for complex connection management.

Scalability: Redis can handle a high number of concurrent connections and messages, making it suitable for systems that need to scale horizontally. Furthermore, Redis can be clustered for even greater scalability.

Performance: Redis is an in-memory data structure store, which allows for fast data access and low latency in message propagation. This is crucial in a live-sharing environment where quick updates and real-time collaboration are essential.

Flexibility: The pub/sub feature in Redis supports pattern matching and filtering, which can be useful for routing messages to the appropriate servers.

Community and support: Redis has a large and active community, which means that there are numerous resources available for troubleshooting, optimization, and customization.

Conclusion:

Overall, the decision to use Redis pub/sub for connecting servers in the live-sharing system was driven by its simplicity, scalability, performance, and flexibility. Additionally, the strong community and support make it a reliable choice for implementing a robust and efficient real-time collaboration platform.

8. System Testing

8.1. Test Automation Framework

8.1.1. Steps for Installing Test Framework

To use the Go testing framework, the Go language must be installed on our operating system. This is the only step that needs to be done since Go testing is integrated within the Go language

[2]. To have Go testing integrated within VS Code, the Go extension must be installed, which will make executing tests much simpler [16].

8.1.2. Steps for Running Test Cases

To run a Go test, a couple of steps are required. First, a Go test file must be created [17]. In these test file(s), the Go test functions can then be written [17]. Since we have installed the Go extension on VS Code, there should be a button above each test function for running it [16].

8.2. Test Case Design

8.2.1. Test Suites

TS-001 ("Java Transpiler File Tests"): A test suite that consists of test cases for the portion of the Java transpiler that deals with parsing Java files. (Refer to **Table 8.2.1** in **Appendix T**)

8.2.2. Unit Test Cases

TC-001 ("Parse File" Unit Test): A unit test case that tests the parseFile function, which will receive a test input of a single project file and should have an expected output including the file name, whether the file's class implements or extends off of another class, variables and methods of the file's class. (Refer to **Table 8.2.2** in **Appendix T**)

TC-002 ("Remove Annotations" Unit Test): A unit test case that tests the removeAnnotations function, which will receive a test input of a file's text, which does not include comments and additional spacing. For the expected output, the file's text should not include comments, spacing, and annotations. (Refer to **Table 8.2.3** in **Appendix T**)

TC-003 ("Remove Comments" Unit Test): A unit test case that tests the removeComments function, which will receive a test input of a file's text and should have an expected output of the file's text without comments. (Refer to **Table 8.2.4** in **Appendix T**)

TC-004 ("Get Package Name" Unit Test): A unit test case that tests the getPackageName function, which will receive a test input of a file's text without comments and should have an expected output of the package name. (Refer to **Table 8.2.5** in **Appendix T**)

TC-005 ("Remove Spacing" Unit Test): A unit test case that tests the removeSpacing function, which will receive a test input of a file's text without comments and should have an expected output of the file's text without comments and spacing. (Refer to **Table 8.2.6** in **Appendix T)**

TC-006 ("Get File Classes" Unit Test): A unit test case that tests the getFileClasses function, which will receive a test input of a file's text with no comments and spacing. For the expected output, there should be an array of all the classes that are defined within the file's text.
(Refer to **Table 8.2.7** in **Appendix T**)

TC-007 ("Get Enum Declarations" Unit Test): A unit test case that tests getEnumDeclarations function, which will receive a test input of a class's inner-text with no comments and spacing. For the expected output, there should be an array of Enum declarations.
(Refer to **Table 8.2.8** in **Appendix T**)

TC-008 ("Get Class Associations" Unit Test): A unit test case that tests getClassAssociations

function, which will receive a test input of an array of class methods and variables. The expected output should include all class type associations such as string, int, byte, etc.
(Refer to **Table 8.2.9** in **Appendix T**)

8.3. Test Case Execution Report

8.3.1. Unit Testing Report

TC-001 Execution Report: Many tests initially failed due to incorrect function logic. After this test case finally passed with a variety of test inputs, we added more functionality to the parseFile function, as we also plan on getting class associations in the function. As a result, our most recent tests have all failed, as we are currently working on obtaining all class associations within the parseFile function. (Refer to **Table 8.3.1** in **Appendix TE**)

TC-002 Execution Report: Many tests initially failed due to incorrect function logic. After debugging and completing the implementation of removeAnnotations, this test case passed with all possible input cases of file text with annotations and produced the correct expected outputs of file text without comments, spacing, and annotations. (Refer to **Table 8.3.2** in **Appendix TE**)

TC-003 Execution Report: Many tests initially failed due to incorrect function logic. After debugging and completing the implementation of removeComments, this test case passed with all possible input cases of file text with comments and produced the correct expected outputs of file text without comments. (Refer to **Table 8.3.3** in **Appendix TE**)

TC-004 Execution Report: Many tests initially failed due to incorrect function logic. After debugging and completing the implementation of getPackageName, this test case passed with all possible input cases of file text without comments and produced the correct expected outputs of package names. (Refer to **Table 8.3.4** in **Appendix TE**)

TC-005 Execution Report: Many tests initially failed due to incorrect function logic. After debugging and completing the implementation of removeSpacing, this test case passed with all possible input cases of file text and produced the correct expected outputs of file text without comments. (Refer to **Table 8.3.5** in **Appendix TE**)

TC-006 Execution Report: Many tests initially failed due to incorrect function logic. After this test case finally passed with a variety of test inputs, we added more functionality to the getFileClasses function, as we also plan on getting class associations in the function. As a result, our most recent tests have failed, as we are currently working on obtaining all class associations within the getFileClasses function. (Refer to **Table 8.3.6** in **Appendix TE**)

TC-007 Execution Report: Many tests initially failed due to incorrect function logic. After debugging and completing the implementation getEnumDeclarations, this test case passed with all possible input cases of a class's inner-text and produced the correct expected outputs of an array with all enum declarations. (Refer to **Table 8.3.7** in **Appendix TE**)

TC-008 Execution Report: Many tests initially failed due to incorrect function logic. We are currently still working on the getClassAssociations function, so our recent tests have all failed. (Refer to **Table 8.3.8** in **Appendix TE**)

9. Challenges and Open Issues

9.1. Challenges Faced in Requirements Engineering

There are many issues that we faced when it came to developing the requirements, documentation, and design of ProUML. For one, it is difficult to work with each other on a consistent basis, due to our conflicting schedules. Additionally, since ProUML is an existing system, we have had to first make sure we understand the existing system before continuing to engineer the system. The existing ProUML system also utilizes libraries and frameworks that some team members have no experience with. So, we also had to make sure we did our research to learn these used libraries and frameworks. Overall, we faced many challenges when engineering our requirements; however, these challenges were dealt with sacrifice, proper communication, review, and research.

10. System Manuals

10.1. Instructions for System Development

10.1.1. How to set up development environment

To set up the development environment for ProUML, many different steps are required. First, the source code must be downloaded from the existing system's GitHub repository. Next, VSCode needs to be installed and opened if it does not yet exist on the development device. After installing/opening VSCode, the necessary programming languages and extensions must be installed, which include Go, Node.js, and Prettier code formatter to ensure GitHub identifies only code changes rather than formatting changes. Then, in a terminal or command line, we need to navigate to the directory, on our development device, where the project is located. Lastly, to run the web server that hosts the ProUML website, the "npm run start" command needs to be run in the terminal or command line.

12. References

[1] PlantUML in a nutshell, *PlantUML.com.* Accessed 10/02/2022,
https://plantuml.com/

[2] Go Documentation, *Documentation.* Accessed 09/15/2022,
https://go.dev/doc/

[3] AntV main, *Liven Data Lively.* Accessed 09/15/2022,
https://antv.vision/en

[4] AntV X6, *X6 JavaScript Diagramming Library.* Accessed 09/15/2022,
https://x6.antv.vision/en

[5] AntV XFlow, *XFlow.* Accessed 09/19/2022,
https://xflow.antv.vision/en-US/

[6] ownCloud, *ownCloud.com.* Accessed 10/12/2022,
https://owncloud.com/

[7] TypeScript Coding Convention, *TypeScript Style Guide.* Accessed 10/17/2022,
https://google.github.io/styleguide/tsguide.html

[8] Git Documentation, *Documentation.* Accessed 09/25/2022,
https://www.git-scm.com/doc

[9] JavaScript Documentation, *JavaScript Reference.* Accessed 09/25/2022,
https://devdocs.io/javascript/

[10] TypeScript Documentation, *Documentation.* Accessed 09/25/2022,
https://www.typescriptlang.org/docs/

[11] React Documentation, *Getting Started.* Accessed 09/25/2022
https://reactjs.org/docs/getting-started.html

[12] Supabase Documentation, *Documentation.* Accessed 09/25/2022
https://supabase.com/docs

[13] Webbee Documentation, *MultiPurpose and UI Kit.* Accessed 09/27/2022
https://webbee.maccarianagency.com/docs-introduction

[14] Fiber Documentation, *Documentation*. Accessed 09/27/2022
https://docs.gofiber.io/

[15] GitHub Documentation, *GitHub Docs*. Accessed 09/15/2022
https://docs.github.com/en

[16] Visual Studio Code Documentation, *Documentation*. Accessed 09/15/2022
https://code.visualstudio.com/docs

[17] How To Write Unit Tests in Go, *Documentation*. Accessed 11/10/2022
https://www.digitalocean.com/community/tutorials/how-to-write-unit-tests-in-go-using-go-test
-and-the-testing-package

[18] Next.js Documentation, *Documentation*. Accessed 12/05/2022

https://nextjs.org/docs/getting-started

[19] Redis Documentation, *Documentation*. Accessed 12/20/2022
https://redis.io/docs/

[20] PostgreSQL Documentation, *Documentation*. Accessed 12/11/2022
https://www.postgresql.org/docs/

[21] Tailwind CSS Documentation, *Documentation*. Accessed 01/03/2023
https://tailwindcss.com/docs/installation

Table 4.1. Use Case Index Table

Project Name: ProUML				
Use Case ID	**Use Case Name**	**Level**	**Author**	**Version**
UC-001	Create UML Diagram	Primary task	Corey Taylor	0.8
UC-002	Translate Between Java and UML	Primary task	Marin Mirasol	0.6
UC-003	Save UML Project	Primary task	Marin Mirasol	0.4
UC-004	Collaborate On ProUML Project	Primary task	Marin Mirasol	0.2
UC-005	Edit UML Diagram	Primary task	Corey Taylor	0.2
Acknowledgment: Generated from the CapStone process management system ©2022				

Table 4.2. Use Case UC-001

Project Name:	**ProUML**
Use Case ID:	UC-001
Use Case Name:	Create UML Diagram
User Goal:	The user should expect the system to allow them to create and edit a ProUML project.
Scope:	ProUML diagram editing system
Level:	Primary task
Relevant User Reqs:	UF-A,UF-B
Relevant System Reqs:	SF-A-01,SF-A-02,SF-A-03,SF-B-01
Primary Actor:	User
Precondition:	Open diagram editor
Minimal Guarantee:	Error messages.
Success Guarantee:	ProUML project created.
Trigger:	When the user creates a new ProUML project with source code, from scratch, or from a provided template.
Success Scenario:	**Step** / **Actions** **1** The user shall either select a template, upload source code, or start a diagram from scratch. **2** The system shall either generate a UML diagram template on the diagram editor, translate from code to UML on the diagram editor, or create an empty diagram on the diagram editor. **3** The user shall be given access to an editable UML diagram on the diagram editor.
Extensions:	**Branching Scenarios**
Acknowledgment: Generated from the CapStone process management system ©2022	

Table 4.3. Use Case UC-002

Project Name:	**ProUML**
Use Case ID:	UC-002
Use Case Name:	Translate Between Java and UML
User Goal:	The user should expect the system to translate Java code to UML diagram form and vice versa.
Scope:	ProUML package parser
Level:	Primary task
Relevant User Reqs:	UF-B
Relevant System Reqs:	SF-B-01,SF-B-02,SF-B-03,SF-C-02,SF-C-03
Primary Actor:	User
Precondition:	Parse the files in the imported package
Minimal Guarantee:	Error messages.
Success Guarantee:	A UML diagram in the diagram editor of their source code or source code from their UML diagram.
Trigger:	When the user imports a Java package.
Success Scenario:	**Step** / **Actions** **1** The user shall import a Java package of their choice. **2** The system shall translate the imported Java code to JSON format by parsing. **3** The system shall generate a UML diagram with the JSON data. **4** The user shall be given access to the generated UML diagram on the diagram editor.
Extensions:	**Branching Scenarios**
Acknowledgment: Generated from the CapStone process management system ©2022	

Table 4.4. Use Case UC-003

Project Name:	**ProUML**
Use Case ID:	UC-003
Use Case Name:	Save UML Project
User Goal:	The user should expect the system to save their ProUML projects.
Scope:	ProUML database storage.
Level:	Primary task
Relevant User Reqs:	UF-C
Relevant System Reqs:	SF-B-03,SF-C-01,SF-C-02,SF-C-03
Primary Actor:	User
Precondition:	Create text file for UML project
Minimal Guarantee:	Error messages.
Success Guarantee:	ProUML project saved.
Trigger:	When a ProUML project is created and edited by the user.
Success Scenario:	**Step** **Actions** **1** The user shall open an existing project or shall create a new project. **2** The user shall make changes to this project. **3** The system shall store these changes in a database. **4** The user shall be able to see these changes reflect on the project when they open the project again on the diagram editor.
Extensions:	**Branching Scenarios**
Acknowledgment: Generated from the CapStone process management system ©2022	

Table 4.5. Use Case UC-004

Project Name:	**ProUML**
Use Case ID:	UC-004
Use Case Name:	Collaborate On ProUML Project
User Goal:	The user should expect the system to allow live collaboration on a ProUML project.
Scope:	ProUML diagram editing system
Level:	Primary task
Relevant User Reqs:	UF-D
Relevant System Reqs:	SF-D-01,SF-D-02
Primary Actor:	User
Precondition:	Communicate to web server
Minimal Guarantee:	Error messages and frozen diagram editor.
Success Guarantee:	Live collaboration with other user(s) on a ProUML project with live updates.
Trigger:	When the user shares their ProUML project to another user.
Success Scenario:	see steps below
Extensions:	**Branching Scenarios**
Acknowledgment: Generated from the CapStone process management system ©2022	

Success Scenario:

Step	**Actions**
1	The user shall open an existing project or shall create a new project.
2	The user shall share this project with another user.
3	The system shall give the added user access the the project owner's project.
4	The user and shared user shall make changes to the project on the diagram editor.
5	The system shall detect all changes made by all users with access to the project via user to server communication.
6	The system shall reflect these changes for all users via server to user communication.
7	The system shall save all changes in real-time.

Table 4.6. Use Case UC-005

Project Name:	**ProUML**
Use Case ID:	UC-005
Use Case Name:	Edit UML Diagram
User Goal:	The user should expect the system to allow them to edit an already-existing diagram.
Scope:	ProUML diagram editing system
Level:	Primary task
Relevant User Reqs:	UF-D
Relevant System Reqs:	SF-A-01,SF-A-02,SF-A-03,SF-C-02,SF-C-03,SF-D-01,SF-D-02
Primary Actor:	User
Precondition:	Open saved ProUML project
Minimal Guarantee:	Error message
Success Guarantee:	ProUML project with the latest saved edits
Trigger:	When a saved ProUML project is opened
Success Scenario:	**Step** / **Actions**: **1** The user shall open an existing project. **2** The user shall make changes to this project on the diagram editor. **3** The system shall reflect these changes on the project by saving any changes made. **4** The user shall see these saved changes if they open the project again.
Extensions:	**Branching Scenarios**
Acknowledgment: Generated from the CapStone process management system ©2022	

Table 4.7. User Functional Requirements: UF-A

Project Name:	**ProUML**				
Requirement #:	**UF-A**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 09:45 AM**		**User**	☒	☐
Modification:	**Sep 27 2022 09:45 AM**		**System**	☐	☐
Description:	Users shall be able to create and edit UML diagrams.				
Priority:	Highest	**✔ High**	Medium	Low	Lowest
This Req. is Refined Into:		**SF-A-01, SF-A-02, SF-A-03**			
Justify why UF-A can be completely covered by SF-A-01, SF-A-02, SF-A-03		To be added later			
Traceability:	**Use cases cf.**	**UC-001**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.8. User Functional Requirements: UF-B

Project Name:	**ProUML**				
Requirement #:	**UF-B**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 09:46 AM**		**User**	☒	☐
Modification:	**Sep 27 2022 09:46 AM**		**System**	☐	☐
Description:	Users shall be able to translate their Java code to UML and vice versa.				
Priority:	Highest	**✔ High**	Medium	Low	Lowest
This Req. is Refined Into:		**SF-B-01, SF-B-02, SF-B-03**			
Justify why UF-B can be completely covered by SF-B-01, SF-B-02, SF-B-03		To be added later			
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-001, TC-002, TC-003, TC-004, TC-005, TC-006, TC-007, TC-008**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.9. User Functional Requirements: UF-C

Project Name:	**ProUML**				
Requirement #:	**UF-C**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 09:46 AM**				
Modification:	**Sep 27 2022 09:46 AM**		**User**	☒	☐
			System	☐	☐
Description:	Users shall be able to save their UML diagram data on ProUML.				
Priority:	Highest	High	[✓ **Medium**]	Low	Lowest
This Req. is Refined Into:		**SF-C-01, SF-C-02, SF-C-03**			
Justify why UF-C can be completely covered by SF-C-01, SF-C-02, SF-C-03		To be added later			
Traceability:	**Use cases cf.**	**UC-003**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.10. User Functional Requirements: UF-D

Project Name:	**ProUML**				
Requirement #:	**UF-D**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 09:46 AM**				
Modification:	**Sep 27 2022 09:47 AM**		**User**	☒	☐
			System	☐	☐
Description:	Users shall be able to collaborate with others on ProUML projects.				
Priority:	Highest	High	Medium	[✓ **Low**]	Lowest
This Req. is Refined Into:		**SF-D-01, SF-D-02**			
Justify why UF-D can be completely covered by SF-D-01, SF-D-02		To be added later			
Traceability:	**Use cases cf.**	**UC-004, UC-005**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.11. User NonFunctional Requirements: UP-01

Project Name:	**ProUML**				
Requirement #:	**UP-01**		**Type**	**Functional**	**Non-Functional**
Creation:	**Oct 02 2022 09:59 PM**		**User**	☐	☒
Modification:	**Oct 26 2022 05:27 PM**		**System**	☐	☐
Description:	Users shall be able to use a web application that is extendable, customizable, and easy to use.		**Product (sub-type below)**		
			Usability Requirements		
Priority:	Highest	High	✓ **Medium**	Low	Lowest
This Req. is Refined Into:		**SP-01-01, SP-01-02, SP-01-03**			
Justify why UP-01 can be completely covered by SP-01-01, SP-01-02, SP-01-03		To be added later			
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.12. System Functional Requirements: SF-A-01

Project Name:	**ProUML**				
Requirement #:	**SF-A-01**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 09:47 AM**		**User**	☐	☐
Modification:	**Sep 27 2022 09:52 AM**		**System**	☒	☐
Description:	The system shall provide a diagram editor designed specifically for UML diagram editing.				
Priority:	✓ **Highest**	High	Medium	Low	Lowest
This Req. is Engineered From:		**UF-A**			
Justify why meeting SF-A-01 can contribute to the fulfilment of UF-A		A UML-based diagram editor helps to reduce user errors since complex logic may be required for the system to comprehend UML syntax.			
Traceability:	**Use cases cf.**	**UC-001, UC-005**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.13. System Functional Requirements: SF-A-02

Project Name:	**ProUML**				
Requirement #:	SF-A-02		**Type**	**Functional**	**Non-Functional**
Creation:	Sep 27 2022 09:51 AM		**User**	☐	☐
Modification:	Sep 27 2022 09:52 AM		**System**	☒	☐
Description:	The system shall provide a button that allows users to add UML class shapes to their UML diagrams.				
Priority:	Highest	High	[✓ Medium]	Low	Lowest
This Req. is Engineered From:		UF-A			
Justify why meeting SF-A-02 can contribute to the fulfilment of UF-A		This gives the users the abilities to add new classes to their UML diagrams.			
Traceability:	**Use cases cf.**	UC-001, UC-005			
	Test cases cf.	Yet to be completed in test case worksheet!			
Acknowledgment	*Generated from the CapStone Process Management System ©2022*				

Table 4.14. System Functional Requirements: SF-A-03

Project Name:	**ProUML**				
Requirement #:	SF-A-03		**Type**	**Functional**	**Non-Functional**
Creation:	Sep 27 2022 09:53 AM		**User**	☐	☐
Modification:	Sep 27 2022 09:54 AM		**System**	☒	☐
Description:	The system shall provide a sidebar for UML class shapes when these shapes are clicked on, where the class name, class associates, whether the class is an interface or not, attributes and methods can be edited by the users.				
Priority:	Highest	High	Medium	Low	Lowest
This Req. is Engineered From:		UF-A			
Justify why meeting SF-A-03 can contribute to the fulfilment of UF-A		This allows users to edit their UML class shapes on their UML diagrams.			
Traceability:	**Use cases cf.**	UC-001, UC-005			
	Test cases cf.	Yet to be completed in test case worksheet!			
Acknowledgment	*Generated from the CapStone Process Management System ©2022*				

Table 4.15. System Functional Requirements: SF-B-01

Project Name:	**ProUML**				
Requirement #:	**SF-B-01**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 09:54 AM**		**User**	☐	☐
Modification:	**Oct 26 2022 05:28 PM**		**System**	☒	☐
Description:	The system shall be able to comprehend Java code into classes, class associations, and class functions by understanding java class names, attributes, and keywords.				
Priority:	Highest	**✓ High**	Medium	Low	Lowest
This Req. is Engineered From:		**UF-B**			
Justify why meeting SF-B-01 can contribute to the fulfilment of UF-B		This helps the system for translating from Java to UML since it understands the needed portions of imported Java code to add into the UML diagrams.			
Traceability:	**Use cases cf.**	**UC-001, UC-002**			
	Test cases cf.	**TC-001, TC-002, TC-003, TC-004, TC-005, TC-006, TC-007, TC-008**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.16. System Functional Requirements: SF-B-02

Project Name:	**ProUML**				
Requirement #:	**SF-B-02**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 09:57 AM**		**User**	☐	☐
Modification:	**Oct 26 2022 05:28 PM**		**System**	☒	☐
Description:	The system shall be able to organize completed UML diagrams into Java code.				
Priority:	Highest	**✓ High**	Medium	Low	Lowest
This Req. is Engineered From:		**UF-B**			
Justify why meeting SF-B-02 can contribute to the fulfilment of UF-B		This helps the system with UML to Java translation since the system know how to create Java code based on the UML diagram information.			
Traceability:	**Use cases cf.**	**UC-002**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.17. System Functional Requirements: SF-B-03

Project Name:	**ProUML**				
Requirement #:	**SF-B-03**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 09:58 AM**		**User**	☐	☐
Modification:	**Nov 08 2022 09:38 AM**		**System**	☒	☐
Description:	The system shall preserve imported code that is not needed in UML diagram translation.				
Priority:	Highest	**✓ High**	Medium	Low	Lowest
This Req. is Engineered From:		**UF-B**			
Justify why meeting SF-B-03 can contribute to the fulfilment of UF-B		This allows imported code, that is not used in the UML diagrams, to be saved, so that users can get the code back when translating from UML to Java.			
Traceability:	**Use cases cf.**	**UC-002, UC-003**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.18. System Functional Requirements: SF-C-01

Project Name:	**ProUML**				
Requirement #:	**SF-C-01**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 10:00 AM**		**User**	☐	☐
Modification:	**Sep 28 2022 11:32 AM**		**System**	☒	☐
Description:	The system shall support user profiles with authentication.				
Priority:	Highest	High	**✓ Medium**	Low	Lowest
This Req. is Engineered From:		**UF-C**			
Justify why meeting SF-C-01 can contribute to the fulfilment of UF-C		This gives the system the ability for user profiles to be saved on a database, where the data associated to specific profiles can be saved.			
Traceability:	**Use cases cf.**	**UC-003**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.19. System Functional Requirements: SF-C-02

Project Name:	**ProUML**				
Requirement #:	**SF-C-02**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 10:02 AM**		**User**	☐	☐
Modification:	**Sep 27 2022 10:02 AM**		**System**	☒	☐
Description:	The system shall save user data and associate the data to specific user profiles in a database.				
Priority:	Highest	High	**✓ Medium**	Low	Lowest
This Req. is Engineered From:		**UF-C**			
Justify why meeting SF-C-02 can contribute to the fulfilment of UF-C		This allows user data to be linked to their specific profile.			
Traceability:	**Use cases cf.**	**UC-002, UC-003, UC-005**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.20. System Functional Requirements: SF-C-03

Project Name:	**ProUML**				
Requirement #:	**SF-C-03**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 10:04 AM**		**User**	☐	☐
Modification:	**Sep 27 2022 10:05 AM**		**System**	☒	☐
Description:	The system shall translate UML diagrams to a text file that can be saved in a database.				
Priority:	Highest	High	**✓ Medium**	Low	Lowest
This Req. is Engineered From:		**UF-C**			
Justify why meeting SF-C-03 can contribute to the fulfilment of UF-C		This allows UML diagram data to be saved into a database.			
Traceability:	**Use cases cf.**	**UC-002, UC-003, UC-005**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.21. System Functional Requirements: SF-D-01

Project Name:	**ProUML**				
Requirement #:	**SF-D-01**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 10:05 AM**		**User**	☐	☐
Modification:	**Sep 27 2022 10:10 AM**		**System**	☒	☐
Description:	The system shall support live sharing by using user-to-user communication.				
Priority:	Highest	**✓ High**	Medium	Low	Lowest
This Req. is Engineered From:		**UF-D**			
Justify why meeting SF-D-01 can contribute to the fulfilment of UF-D		This allows the same shared ProUML project to update as changes are made by both users.			
Traceability:	**Use cases cf.**	**UC-004, UC-005**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.22. System Functional Requirements: SF-D-02

Project Name:	**ProUML**				
Requirement #:	**SF-D-02**		**Type**	**Functional**	**Non-Functional**
Creation:	**Sep 27 2022 10:08 AM**		**User**	☐	☐
Modification:	**Nov 08 2022 09:40 AM**		**System**	☒	☐
Description:	The system shall be able to save collaborated projects in real time.				
Priority:	Highest	High	Medium	Low	Lowest
This Req. is Engineered From:		**UF-D**			
Justify why meeting SF-D-02 can contribute to the fulfilment of UF-D		This allows the system to automatically save and update shared ProUML projects between collaborators.			
Traceability:	**Use cases cf.**	**UC-004, UC-005**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.23. System NonFunctional Requirements: SP-01-01

Project Name:	**ProUML**				
Requirement #:	**SP-01-01**		**Type**	**Functional**	**Non-Functional**
Creation:	**Oct 02 2022 10:00 PM**		**User**	☐	☐
Modification:	**Oct 02 2022 10:02 PM**		**System**	☐	☒
Description:	The system shall provide a user login page where users can log into their accounts.		**Product (sub-type below)**		
			Usability Requirements		
Priority:	Highest	**✓ High**	Medium	Low	Lowest
This Req. is Engineered From:		**UP-01**			
Justify why meeting SP-01-01 can contribute to the fulfilment of UP-01		A user account can help to organize user data into specific user profiles.			
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.24. System NonFunctional Requirements: SP-01-02

Project Name:	**ProUML**				
Requirement #:	**SP-01-02**		**Type**	**Functional**	**Non-Functional**
Creation:	**Oct 02 2022 10:02 PM**		**User**	☐	☐
Modification:	**Oct 02 2022 10:04 PM**		**System**	☐	☒
Description:	The system shall provide a dashboard page with the ability to import source code to translate to Java code, start a UML diagram from scratch, or choose a template design pattern diagram to start from.		**Product (sub-type below)**		
			Usability Requirements		
Priority:	Highest	High	**✓ Medium**	Low	Lowest
This Req. is Engineered From:		**UP-01**			
Justify why meeting SP-01-02 can contribute to the fulfilment of UP-01		A dashboard page will allow the organization of important and commonly utilized features on one page.			
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Table 4.25. System NonFunctional Requirements: SP-01-03

Project Name:	**ProUML**				
Requirement #:	**SP-01-03**				
Creation:	**Oct 02 2022 10:04 PM**				
Modification:	**Oct 02 2022 10:05 PM**				
Description:	The system shall allow users to edit the user interface with dark and light modes.			**Product (sub-type below)** **Usability Requirements**	
Priority:	Highest	High	Medium	Low	**✓ Lowest**
This Req. is Engineered From:		**UP-01**			
Justify why meeting SP-01-03 can contribute to the fulfilment of UP-01		This allows users to customize the user interface of the web application to their liking.			
Traceability:	**Use cases cf.**	**N/A**			
	Test cases cf.	**Yet to be completed in test case worksheet!**			
Acknowledgment	***Generated from the CapStone Process Management System ©2022***				

Type	**Functional**	**Non-Functional**
User	☐	☐
System	☐	☒

Table 4.26. Mapping from user requirements to system requirements

Project Name: ProUML			
User Requirements		**System Requirements**	
Req ID	**Description**	**Req ID**	**Description**
UF-A	Users shall be able to create and edit UML diagrams.	SF-A-01	The system shall provide a diagram editor designed specifically for UML diagram editing.
		SF-A-02	The system shall provide a button that allows users to add UML class shapes to their UML diagrams.
		SF-A-03	The system shall provide a sidebar for UML class shapes when these shapes are clicked on, where the class name, class associates, whether the class is an interface or not, attributes and methods can be edited by the users.
UF-B	Users shall be able to translate their Java code to UML and vice versa.	SF-B-01	The system shall be able to comprehend Java code into classes, class associations, and class functions by understanding java class names, attributes, and keywords.
		SF-B-02	The system shall be able to organize completed UML diagrams into Java code.
		SF-B-03	The system shall preserve imported code that is not needed in UML diagram translation.
UF-C	Users shall be able to save their UML diagram data on ProUML.	SF-C-01	The system shall support user profiles with authentication.
		SF-C-02	The system shall save user data and associate the data to specific user profiles in a database.
		SF-C-03	The system shall translate UML diagrams to a text file that can be saved in a database.
UF-D	Users shall be able to collaborate with others on ProUML projects.	SF-D-01	The system shall support live sharing by using user-to-user communication.
		SF-D-02	The system shall be able to save collaborated projects in real time.
UP-01	Users shall be able to use a web application that is extendable, customizable, and easy to use.	SP-01-01	The system shall provide a user login page where users can log into their accounts.
		SP-01-02	The system shall provide a dashboard page with the ability to import source code to translate to Java code, start a UML diagram from scratch, or choose a template design pattern diagram to start from.
		SP-01-03	The system shall allow users to edit the user interface with dark and light modes.
Acknowledgment: Generated from the CapStone process management system ©2022			

Table 8.2.1. Test Suite TS-001: Java Transpiler File Tests

Test Case ID	Test Stage	Test Case Description	Tested
TC-001	Unit	Parse File	Yes
TC-002	Unit	Remove Annotations	Yes
TC-003	Unit	Remove Comments	Yes
TC-004	Unit	Get Package Name	Yes
TC-005	Unit	Remove Spacing	Yes
TC-006	Unit	Get File Classes	Yes
TC-007	Unit	Get Enum Declarations	Yes
TC-008	Unit	Get Class Relation Types	Yes

Table 8.2.2. Test Case TC-001

Project Name:	**ProUML**	
Test Suite	TS-001: Java Transpiler File Tests	
Test Case ID	TC-001 (Unit Test)	
What To Test	Parse File	
Test Data Input	Project file	
Expected Result	File name, implements, extends, variables, methods	
Traceability	**Relevant User Req.(s)**	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2022		

Table 8.2.3. Test Case TC-002

Project Name:	**ProUML**	
Test Suite	TS-001: Java Transpiler File Tests	
Test Case ID	TC-002 (Unit Test)	
What To Test	Remove Annotations	
Test Data Input	File text (no comments and no spacing)	
Expected Result	File text (no comments, no spacing, no annotations)	
Traceability	**Relevant User Req.(s)**	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2022		

Table 8.2.4. Test Case TC-003

Project Name:	**ProUML**	
Test Suite	TS-001: Java Transpiler File Tests	
Test Case ID	TC-003 (Unit Test)	
What To Test	Remove Comments	
Test Data Input	File text	
Expected Result	File text (no comments)	
Traceability	**Relevant User Req.(s)**	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2022		

Table 8.2.5. Test Case TC-004

Project Name:	**ProUML**	
Test Suite	TS-001: Java Transpiler File Tests	
Test Case ID	TC-004 (Unit Test)	
What To Test	Get Package Name	
Test Data Input	File text (no comments)	
Expected Result	Package name	
Traceability	**Relevant User Req.(s)**	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2022		

Table 8.2.6. Test Case TC-005

Project Name:	**ProUML**	
Test Suite	TS-001: Java Transpiler File Tests	
Test Case ID	TC-005 (Unit Test)	
What To Test	Remove Spacing	
Test Data Input	File text (no comments)	
Expected Result	File text (no comments and no spacing)	
Traceability	**Relevant User Req.(s)**	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2022		

Table 8.2.7. Test Case TC-006

Project Name:	**ProUML**	
Test Suite	TS-001: Java Transpiler File Tests	
Test Case ID	TC-006 (Unit Test)	
What To Test	Get File Classes	
Test Data Input	File text (no comments and no spacing)	
Expected Result	Array of classes	
Traceability	**Relevant User Req.(s)**	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2022		

Table 8.2.8. Test Case TC-007

Project Name:	**ProUML**	
Test Suite	TS-001: Java Transpiler File Tests	
Test Case ID	TC-007 (Unit Test)	
What To Test	Get Enum Declarations	
Test Data Input	Enum class inner-text (no comments and no spacing)	
Expected Result	Array of declarations	
Traceability	**Relevant User Req.(s)**	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2022		

Table 8.2.9. Test Case TC-008

Project Name:	**ProUML**	
Test Suite	TS-001: Java Transpiler File Tests	
Test Case ID	TC-008 (Unit Test)	
What To Test	Get Class Relation Types	
Test Data Input	Array of class methods and variables	
Expected Result	Class type associations and dependencies (includes all types such as String, int, byte, etc.)	
Traceability	**Relevant User Req.(s)**	UF-B
	Relevant System Req.(s)	SF-B-01
	Relevant Use Case(s)	UC-002
Acknowledgment: Generated from the CapStone process management system ©2022		

Table 8.3.1. Execution Report of Test Case TC-001

Project Name:	**ProUML**
Test Case ID:	TC-001
Testing Tools Used:	Go Testing
Testing Type:	Functional testing
Execution Steps:	1 cd backend/transpiler/java 2 go test -run TestParseFile

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	All Team Members	10/23/2022	N/A	Fail		
2	All Team Members	10/25/2022	N/A	Fail		
3	All Team Members	10/29/2022	N/A	Fail		
4	All Team Members	10/30/2022	N/A	Pass		
5	All Team Members	11/02/2022	N/A	Pass		
6	All Team Members	11/05/2022	N/A	Pass		
7	All Team Members	11/08/2022	N/A	Pass		
8	All Team Members	11/12/2022	N/A	Fail	Added class associations into expected return value. Not yet implemented into code.	
9	All Team Members	11/16/2022	N/A	Fail		
10	All Team Members	11/21/2022	N/A	Fail		
11	All Team Members	11/28/2022	N/A	Fail		

Execution Summary:	Many tests initially failed due to incorrect function logic. After this test case finally passed with a variety of test inputs, we added more functionality to the parseFile function, as we also plan on getting class associations in the function. As a result, our most recent tests have all failed, as we are currently working on obtaining all class associations within the parseFile function.

Table 8.3.2. Execution Report of Test Case TC-002

Project Name:	**ProUML**
Test Case ID:	TC-002
Testing Tools Used:	Go Testing
Testing Type:	Functional testing
Execution Steps:	1 cd backend/transpiler/java 2 go test -run TestRemoveAnnotations

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	All Team Members	10/23/2022	N/A	Fail		
2	All Team Members	10/25/2022	N/A	Fail		
3	All Team Members	10/29/2022	N/A	Pass		
4	All Team Members	10/30/2022	N/A	Pass		
5	All Team Members	11/02/2022	N/A	Pass		
6	All Team Members	11/05/2022	N/A	Pass		
7	All Team Members	11/08/2022	N/A	Pass		
8	All Team Members	11/12/2022	N/A	Pass		
9	All Team Members	11/16/2022	N/A	Pass		
10	All Team Members	11/21/2022	N/A	Pass		
11	All Team Members	11/28/2022	N/A	Pass		

Execution Summary:	Many tests initially failed due to incorrect function logic. After debugging and completing the implementation of removeAnnotations, this test case passed with all possible input cases of file text with annotations and produced the correct expected outputs of file text without comments, spacing, and annotations.

Table 8.3.3. Execution Report of Test Case TC-003

Project Name:	**ProUML**
Test Case ID:	TC-003
Testing Tools Used:	Go Testing
Testing Type:	Functional testing
Execution Steps:	1 cd backend/transpiler/java 2 go test -run TestRemoveComments

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	All Team Members	10/30/2022	N/A	Fail		
2	All Team Members	11/02/2022	N/A	Fail		
3	All Team Members	11/05/2022	N/A	Pass		
4	All Team Members	11/08/2022	N/A	Pass		
5	All Team Members	11/12/2022	N/A	Pass		
6	All Team Members	11/16/2022	N/A	Pass		
7	All Team Members	11/21/2022	N/A	Pass		
8	All Team Members	11/28/2022	N/A	Pass		

Execution Summary:	Many tests initially failed due to incorrect function logic. After debugging and completing the implementation of removeComments, this test case passed with all possible input cases of file text with comments and produced the correct expected outputs of file text without comments.

Acknowledgment: Generated from the CapStone process management system ©2022

Table 8.3.4. Execution Report of Test Case TC-004

Project Name:	**ProUML**
Test Case ID:	TC-004
Testing Tools Used:	Go Testing
Testing Type:	Functional testing
Execution Steps:	1 cd backend/transpiler/java 2 go test -run TestGetPackageName

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	All Team Members	10/30/2022	N/A	Fail		
2	All Team Members	11/02/2022	N/A	Fail		
3	All Team Members	11/05/2022	N/A	Pass		
4	All Team Members	11/08/2022	N/A	Pass		
5	All Team Members	11/12/2022	N/A	Pass		
6	All Team Members	11/16/2022	N/A	Pass		
7	All Team Members	11/21/2022	N/A	Pass		
8	All Team Members	11/28/2022	N/A	Pass		

Execution Summary:	Many tests initially failed due to incorrect function logic. After debugging and completing the implementation of getPackageName, this test case passed with all possible input cases of file text without comments and produced the correct expected outputs of package names.

Acknowledgment: Generated from the CapStone process management system ©2022

Table 8.3.5. Execution Report of Test Case TC-005

Project Name:	**ProUML**
Test Case ID:	TC-005
Testing Tools Used:	Go Testing
Testing Type:	Functional testing
Execution Steps:	1 cd backend/transpiler/java 2 go test -run TestRemoveSpacing

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	All Team Members	10/23/2022	N/A	Fail		
2	All Team Members	10/25/2022	N/A	Fail		
3	All Team Members	10/29/2022	N/A	Fail		
4	All Team Members	10/30/2022	N/A	Fail		
5	All Team Members	11/02/2022	N/A	Fail		
6	All Team Members	11/05/2022	N/A	Pass		
7	All Team Members	11/08/2022	N/A	Pass		
8	All Team Members	11/12/2022	N/A	Pass		
9	All Team Members	11/16/2022	N/A	Pass		
10	All Team Members	11/21/2022	N/A	Pass		
11	All Team Members	11/28/2022	N/A	Pass		

Execution Summary:	Many tests initially failed due to incorrect function logic. After debugging and completing the implementation of removeSpacing, this test case passed with all possible input cases of file text and produced the correct expected outputs of file text without comments.

Acknowledgment: Generated from the CapStone process management system ©2022

Table 8.3.6. Execution Report of Test Case TC-006

Project Name:	**ProUML**
Test Case ID:	TC-006
Testing Tools Used:	Go Testing
Testing Type:	Functional testing
Execution Steps:	1 cd backend/transpiler/java 2 go test -run TestGetFileClasses

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	All Team Members	11/02/2022	N/A	Fail		
2	All Team Members	11/05/2022	N/A	Fail		
3	All Team Members	11/08/2022	N/A	Pass		
4	All Team Members	11/12/2022	N/A	Pass		
5	All Team Members	11/16/2022	N/A	Pass		
6	All Team Members	11/21/2022	N/A	Fail	Added class associations into expected return value. Not yet implemented into code.	
7	All Team Members	11/28/2022	N/A	Fail		

Execution Summary:	Many tests initially failed due to incorrect function logic. After this test case finally passed with a variety of test inputs, we added more functionality to the getFileClasses function, as we also plan on getting class associations in the function. As a result, our most recent tests have failed, as we are currently working on obtaining all class associations within the getFileClasses function.

Acknowledgment: Generated from the CapStone process management system ©2022

Table 8.3.7. Execution Report of Test Case TC-007

Project Name:	**ProUML**
Test Case ID:	TC-007
Testing Tools Used:	Go Testing
Testing Type:	Functional testing
Execution Steps:	1 cd backend/transpiler/java 2 go test -run TestGetEnumDeclarations

Test Execution Records:

#	Tester	Test Date	Actual Result	Status	Defect	Correction
1	All Team Members	11/08/2022	N/A	Fail		
2	All Team Members	11/12/2022	N/A	Fail		
3	All Team Members	11/16/2022	N/A	Pass		
4	All Team Members	11/21/2022	N/A	Pass		
5	All Team Members	11/28/2022	N/A	Pass		

Execution Summary:	Many tests initially failed due to incorrect function logic. After debugging and completing the implementation getEnumDeclarations, this test case passed with all possible input cases of a class's inner-text and produced the correct expected outputs of an array with all enum declarations.

Acknowledgment: Generated from the CapStone process management system ©2022

Table 8.3.8. Execution Report of Test Case TC-008

Project Name:	**ProUML**					
Test Case ID:	TC-008					
Testing Tools Used:	Go Testing					
Testing Type:	Functional testing					
Execution Steps:	1	cd backend/transpiler/java				
	2	go test -run TestGetClassRelationTypes				
Test Execution Records:						
#	**Tester**	**Test Date**	**Actual Result**	**Status**	**Defect**	**Correction**
1	All Team Members	11/12/2022	N/A	Fail		
2	All Team Members	11/16/2022	N/A	Fail		
3	All Team Members	11/21/2022	N/A	Fail		
4	All Team Members	11/28/2022	N/A	Fail		
Execution Summary:	Many tests initially failed due to incorrect function logic. We are currently still working on the getClassRelationTypes function, so our recent tests have all failed.					
Acknowledgment: Generated from the CapStone process management system ©2022						