Project Title

Project Team

Student 1 19I-1234
Student 2 19I-1234
Student 3 19I-1234

Session 2018-2022

Supervised by

Mr./ Ms./ Dr. Supervisor Name

Co-Supervised by

Mr./ Ms./ Dr. Supervisor Name

Department of Computer Science

National University of Computer and Emerging Sciences
Islamabad, Pakistan

June, 2022

Contents

1 Introduction [AS PER SCOPE DOCUMENT] 1

1.1 Existing Solutions . . . 2
1.2 Problem Statement . . . 3
1.3 Scope . . . 4
1.4 Modules . . . 4
1.4.1 User Interface (UI) . . . 4
1.4.2 User Management . . . 5
1.4.3 Authentication and Authorization . . . 5
1.4.4 Security and Privacy . . . 5
1.4.5 Version Control and Data Backup . . . 5
1.4.6 Real time Synchronization . . . 5
1.4.7 Tutor Matching Based on Student Preferences . . . 6
1.4.8 Video calling . . . 6
1.4.9 Real time emotional analytics . . . 6
1.4.10 Social Learning Communities . . . 6
1.4.11 Report Generation . . . 6
1.4.12 Feedback . . . 7
1.5 Work Division . . . 7

2 Project Requirements [AS PER FYP1 MID REPORT] 9

2.1 Use-case . . . 9
2.1.1 Use Case 1: Register Student . . . 10
2.1.2 Use Case 2: Create Profile for Tutor . . . 11
2.1.3 Use Case 3 : Search for Tutor . . . 12
2.1.4 Use Case 4: Book a Tutoring Session . . . 13
2.1.5 Use Case 5: Attend Tutoring Session . . . 13
2.1.6 Use Case 6: Monitor Real-Time Emotional Analytics . . . 14
2.1.7 Use Case 7: Generate Progress Report . . . 15
2.1.8 Use Case 8: Provide Feedback . . . 16
2.1.9 Use Case 9: Join Social Learning Communities . . . 16
2.1.10 Use Case 10: Manage Accounts . . . 17
2.1.11 Use Case 11: Monitor Platform Activity . . . 18

2.1.12 Use Case 12: Manage Payments 18
2.1.13 Use Case 13: Send Notification 19
2.2 Functional Requirements 20
2.2.1 User Interface (UI) 20
2.2.2 User Management 20
2.2.3 Authentication and Authorization 20
2.2.4 Security and Privacy 21
2.2.5 Version Control and Data Backup 21
2.2.6 Real-time Synchronization 21
2.2.7 Tutor Matching Based on Student Preferences 21
2.2.8 Video Calling 22
2.2.9 Real-time Emotional Analytics 22
2.2.10 Social Learning Communities 22
2.2.11 Report Generation 22
2.2.12 Feedback 23
2.3 Non-Functional Requirements 23
2.3.1 Reliability 23
2.3.2 Usability 24
2.3.3 Performance 25
2.3.4 Security 25

3 System Overview [AS PER FYP1 MID REPORT] 27
3.1 Architectural Design 27
3.2 Data Design 29
3.2.1 Major Data Entities 29
3.2.1.1 Users 29
3.2.1.2 TutorProfiles 29
3.2.1.3 StudentProfiles 29
3.2.1.4 ParentProfiles 30
3.2.1.5 AdminProfiles 30
3.2.1.6 Sessions 30
3.2.1.7 Feedback 30
3.2.1.8 Reports 30
3.2.1.9 EmotionalAnalytics 31
3.2.1.10 Payments 31
3.2.1.11 Preferences 31
3.2.2 Data Storage and Organization 31
3.2.3 Data Processing 32
3.2.4 Data Security 32
3.3 Domain Model 33

3.4 Design Models . . . 34
3.4.1 Activity Diagram . . . 35
3.4.2 Class Diagram . . . 36
3.4.3 Sequence Diagram . . . 37
3.4.3.1 **SSD 1: Student Register** . . . 38
3.4.3.2 **SSD 2: Tutor Profile Creation** . . . 39
3.4.3.3 **SSD 3: Search For Tutor** . . . 40
3.4.3.4 **SSD 4: Book Tutoring session** . . . 41
3.4.3.5 **SSD 5: Attend Session** . . . 42
3.4.3.6 **SSD 6: Emotional Analytics** . . . 43
3.4.3.7 **SSD 7: Generate Progress Report** . . . 44
3.4.3.8 **SSD 8: Feedback** . . . 45
3.4.3.9 **SSD 9: Join Communities** . . . 46
3.4.3.10 **SSD 10: Manage Account** . . . 47
3.4.3.11 **SSD 11: Monitor Platform Activity** . . . 48
3.4.3.12 **SSD 12: Manage Payment** . . . 49
3.4.3.13 **SSD 13: Send Notification** . . . 50
3.4.3.14 **SSD 14: User Login** . . . 51
3.4.4 State Transition Diagram . . . 52

4 Implementation and Testing [UPTO THE CURRENT ITERATION ONLY] 53
4.1 Algorithm Design . . . 53
4.2 External APIs/SDKs . . . 54
4.3 Testing Details . . . 54
4.3.1 Unit Testing . . . 54

References 55

A Appendices 57
A.1 Appendix A . . . 57
A.1.1 Use Case Diagram example (Online Shopping System) . . . 57
A.1.2 Detail Use Case Example . . . 58
A.1.3 Event-Response Table for a Highway Intersection . . . 59
A.1.4 Story Board Example For Android App . . . 60
A.2 Appendix B . . . 61
A.2.1 Domain Model Example For Online Shopping . . . 61
A.3 Appendix C . . . 62
A.3.1 Box And Line Example For Online Shopping . . . 62
A.3.2 Architecture Pattern Example For Online Shopping . . . 63
A.4 Appendix D . . . 64
A.4.1 Activity Diagram . . . 64

A.4.2 Class Diagram 65
A.4.3 Sequence Diagram 66
A.4.4 State Transition Diagram 66
A.4.5 Data Flow Diagram 67

List of Figures

2.1 Use Case Diagram 10

3.1 Architecture Diagram 28
3.2 Data Design Diagram 33
3.3 Domain Model 34
3.4 Activity Diagram 36
3.5 Class Diagram 37
3.6 SSD (Student Register) 38
3.7 SSD (Tutor Profile Creation) 39
3.8 SSD (Search For Tutor) 40
3.9 SSD (Book Tutoring session) 41
3.10 SSD (Attend Session) 42
3.11 SSD (Emotional Analytics) 43
3.12 SSD (Generate Progress Report) 44
3.13 SSD (Feedback) 45
3.14 SSD (Join Communities) 46
3.15 SSD (Manage Account) 47
3.16 SSD (Monitor Platform Activity) 48
3.17 SSD (Manage Payment) 49
3.18 SSD (Send Notification) 50
3.19 SSD (User Login) 51
3.20 State Diagram 52

4.1 Example Of Algorithm Design 53
4.2 Example for Unit Testing 54

A.1 Use Case Diagram for the Online Shopping System 57
A.2 Detail Use Case Example 58
A.3 Example of Event Response Table 59
A.4 Example of Story Board 60
A.5 Domain Model Example For Online Shopping Application 61
A.6 Box and Line Diagram For Online Shopping Application 62
A.7 Architecture Pattern For Online Shopping Application 63
A.8 Activity Diagram For Online Shopping Application 64

A.9 Class Diagram For Online Shopping Application 65
A.10 Sequence Diagram For Online Shopping Application 66
A.11 State Transition Diagram For Online Shopping Application 66

List of Tables

1.1 Comparison of Existing Solutions 3
1.2 Table 1 7

Chapter 1

Introduction [AS PER SCOPE DOCUMENT]

The need for personalized and interactive learning methods has grown significantly in the world today. In order to provide an engaging and efficient learning environment, the SmartTutor combines modern technology including real-time emotional analytics, and video communication.

The purpose of this project is to provide a web-based platform with functionality like feedback systems, social learning communities, video calling, and tutor-student matching based on preferences that enable smooth tutor-student interactions. Via this platform, the facial emotions of the students will be captured in real time in a video call session between tutor and student, which will allow tutor to adjust their teaching strategies.

Background:

The education sector has witnessed rapid evolution over the past decade, driven by technological advancements aimed at improving learning experiences. However, despite the availability of online learning platforms and digital tools, personalized and engaging learning remains a significant challenge. Traditional education systems often fail to cater to individual student needs, leading to disengagement and suboptimal learning outcomes. Additionally, the lack of real-time feedback mechanisms prevents tutors from dynamically adjusting their teaching strategies based on student engagement or emotional response during sessions.

One key issue is the difficulty students face in finding tutors whose teaching styles align with their learning preferences. Generic tutor-student matching systems often rely on superficial criteria, such as subject expertise or availability, neglecting factors like teaching methodology, student personality, and emotional compatibility. This mismatch frequently results in less effective learning sessions.

Moreover, existing systems have limited parental involvement in a child's academic jour-

ney. Many platforms lack mechanisms to provide detailed progress tracking, emotional engagement analytics, or collaborative features for tutors, students, and parents to work together in fostering academic improvement.

Several platforms have attempted to address these issues. For instance, systems like smarttutor.co. offered online tutor-student pairing but lacked real-time emotional analytics, which are crucial for improving session interactivity and effectiveness. Similarly, research studies emphasized the importance of adaptive learning but failed to integrate technological solutions for emotional engagement tracking into live sessions.

Recognizing these gaps, SmartTutor was conceived as an innovative solution to create a comprehensive, user-friendly platform that addresses these challenges. By integrating real-time emotional analytics, personalized tutor matching, and collaborative tools for all stakeholders, SmartTutor seeks to redefine the online learning landscape and foster a more engaging, effective educational experience.

[1].

1.1 Existing Solutions

Over the years, several platforms have emerged to address the challenges in education, particularly in connecting students with tutors. While these systems provide valuable services, they often fall short of delivering comprehensive and personalized learning experiences. Below is an overview of some of the existing solutions, their features, and their limitations.

Table 1.1: Comparison of Existing Solutions

System Name	System Overview	System Limitations
Beacon Tutoring Pakistan	Beacon Tutoring Pakistan connects students with tutors based on location, subject preference, and availability. The platform focuses on creating an optimal tutor-student match to ensure academic progress.	Lacks advanced features like video calling or emotional analytics to monitor student engagement. Primarily focuses on basic tutor-student matching, limiting its scope in fostering interactive and adaptive learning.
Preply	Preply is a global platform that facilitates tutor-student connections for online learning. It offers search filters such as subject expertise, hourly rates, and tutor language.	Does not incorporate emotional analytics to gauge student engagement or provide feedback. Also lacks collaborative tools for parents and tutors to monitor progress.
SmartTutor (Proposed)	SmartTutor combines real-time emotional analytics, personalized tutor matching, video calling, and detailed progress tracking. The system enables dynamic tutor-student interaction while involving parents in the learning process.	The system doesn't have such limitations. This system provides a better environment for students to find a best tutor for themselves and align with them.

This comparison highlights the gaps in existing solutions, such as the absence of emotional analytics and real-time feedback mechanisms, and establishes the need for a more advanced platform like *SmartTutor*.

1.2 Problem Statement

In today's educational environment, finding a right tutor for student according to their preferences is still very difficult. This process also lacks personalization which leads to a mismatch between the learning style of student and the teaching style of tutor. This difficulty in finding a right tutor results in less effective tutoring sessions which results in an outcome which is not improved or is less effective.

Moreover, traditional education system lacks in **student engagement** which also fails to provide an interactive learning environment. Without knowing that how the students are reacting to the lecture delivered to them, tutors are unable to adjust their teaching strategies.

Inadequate involvement of parents is also one of the major problem which leads to an unimproved progress of a student. This disconnect between the tutor and the parent result in an uninformed progress of a child in front of his/her parents which also leads to a bad student progress.

The SmartTutor: An Enhanced Learning System seeks to overcome these challenges by giving a more individualized, engaging, and inclusive tutoring experience.

1.3 Scope

SmartTutor is a platform that provides effective matching of tutor based on the student's preferences. Tutors can list their qualifications, skills, availability, and areas of expertise by creating their profiles, whereas the students can search and select the best tutor according to their requirements. It also features Video Calling integrated with real time emotional analytics that allow the teachers to monitor the student's engagement levels during live sessions. Additionally, it creates social learning communities where students can collaborate and share knowledge. It also generates detailed reports to monitor both teacher and students performance, which helps in improving outcomes. This collaborative learning will keep parents connected with their children.

1.4 Modules

Following are the modules and features that SmartTutor consist of:

1.4.1 User Interface (UI)

It involves designing a user friendly and responsive interface for both tutor and students for their ease.

1. User **friendly design** using different frameworks i.e react.

2. A **responsive design** that works for a different devices i.e mobile, laptop.

1.4.2 User Management

This module will help admin to manage users i.e student, tutor, parent profiles.

1. **Admin dashboard** where admin can **manage accounts** of different users.
2. This will also help admin to **track and review** the activity of users.

1.4.3 Authentication and Authorization

This module will allow user to make a successful register or login to their accout that will also ensure that authorizd users have access to the website.

1. **Secure authentication** for end users.
2. Role based access control to ensure specific user functionalities.

1.4.4 Security and Privacy

This module ensures that all the data of users is safe.

1. **End to end encryption** in video calling.
2. **Data encryption** to ensure security of data.

1.4.5 Version Control and Data Backup

This module will ensure that the data generated is protected and is not lossed.

1. **Data backups**.
2. **Data redundancy**.

1.4.6 Real time Synchronization

This module will ensure real time communication between users.

1. **Real time synchronization** for video calling and emotional analytics.

1.4.7 Tutor Matching Based on Student Preferences

This module allows students to find the best tutor based on personalized preferences, ensuring an optimized tutor-student pairing.

1. **Preference based finding** on bases of subject, availability etc.
2. **Tutor profiles** including their qualifications, experience etc.

1.4.8 Video calling

This module allows a secure video calling environment where students and tutors can interact with each other.

1. **Video and audio** streaming for smooth communication.
2. **Screen sharing** option for both student and tutors.

1.4.9 Real time emotional analytics

This module will catch the emotional state of students during video calling which will help tutors to adjust their teaching methods.

1. **Emotion detection** to identify emotions like satisfaction, confusion etc.
2. **Real-time emotional** feedback for tutors to adjust their teaching strategies.

1.4.10 Social Learning Communities

This module will create social learning communities in which students can collaborate and interact

1. **Group Creation** for a learning environment which is collaborative.
2. **Shared Resources Repository** where user can send helping material.

1.4.11 Report Generation

This module will give a report to parents about their child's learning progress while tutoring sessions.

1. **Progress Reports** where reports of student progress is displayed to parent.
2. **Exportable Reports** in PDF or png format, which can be downloaded by parents .

1.4.12 Feedback

This module will facilitate parents and students to provide a feedback to a particular tutor.

1. **Feedback Form** to provide an anonymous feedback for tutor.

1.5 Work Division

For each module and respective feature, assign responsibility to a team member.

Table 1.2: Table 1

Name	**Registration**	**Responsibility / Module / Feature**
Alian Anwar	21i-0730	Module 1: User Interface (UI) - Iterations 1-4 Module 3: Authentication and Authorization - Iteration 2 Module 6: Real-Time Synchronization - Iteration 3 Module 7: Tutor Matching Based on Student Preferences - Iteration 2 Module 12: Feedback and Rating System - Iterations 3-4 Testing - Iterations 1-4
Munam	21i-0460	Module 2: User Management - Iterations 1-2 Module 4: Security and Privacy - Iteration 2 Module 8: Video Calling - Iterations 3-4 Module 10: Social Learning Communities - Iterations 3-4 Module 12: Feedback and Rating System - Iterations 3-4 Testing - Iterations 1-4
Raza	21i-0511	Module 5: Version Control and Data Backup - Iterations 1-2 Module 9: Real-Time Emotional Analytics - Iterations 2-3 Module 11: Report Generation - Iterations 2-3 Module 12: Feedback and Rating System - Iterations 3-4 Testing - Iterations 1-4

Chapter 2

Project Requirements [AS PER FYP1 MID REPORT]

This section outlines the necessary requirements for the successful completion of the project. Project requirements can be divided into two main categories: functional requirements, which describe the system's core operations, and non-functional requirements, which specify performance, security, and usability standards.

2.1 Use-case

This section outlines in detail the major use cases offered on the SmartTutor platform. A use case is a description of how users and the system interact to achieve some goals. It specifies who the actors are, the flow of events, and the possible terminations that may be brought about. These use cases describe our intended scenario, which in turn ensures that the system behaves correspondingly for various scenarios, thus meeting the functional requirements defined for each user class.

The following are some use cases that describe the core functionalities of the SmartTutor platform, like user sign-up, interaction between tutor and student, video calling, report generation about progress, and real-time emotional analytics etc.

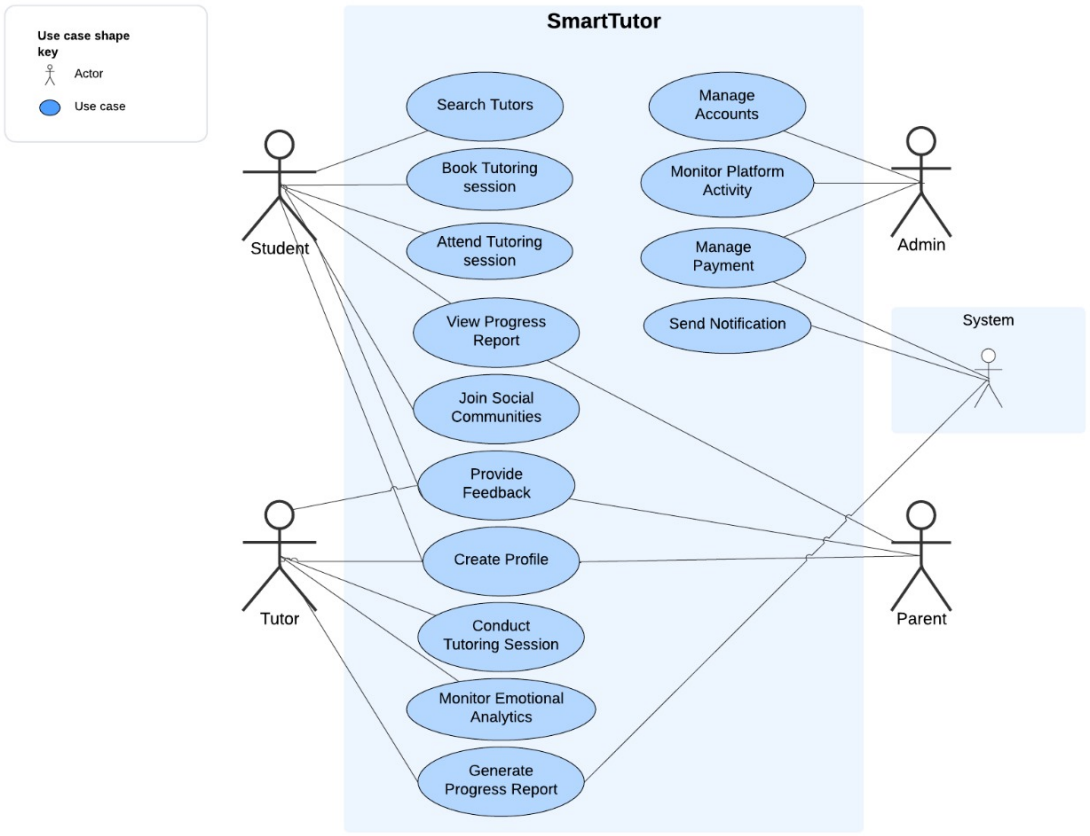

Figure 2.1: Use Case Diagram

2.1.1 Use Case 1: Register Student

- **Actor:** Student
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Precondition:**
  1. Students should have a valid email address or phone number.
  2. Students have opened the sign-up page.
- **Success Guarantee:**
  1. The student's account is successfully created and the student can now log in.
- **Main Success Scenario:**

1. The student goes to the registration page.
2. The student enters their personal details, email, and password.
3. The system validates the email and password.
4. The system checks for duplicate emails.
5. The system confirms the verification and completes the registration process.

- **Extension:**

1. The student enters an invalid email address or password format.
2. The system displays that the email is already registered.
3. The system fails to verify the email.

2.1.2 Use Case 2: Create Profile for Tutor

- **Actor:** Tutor
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Preconditions:**

1. The tutor is already registered and successfully logged into the platform.
2. The tutor has selected the edit or create profile option.

- **Success Guarantee:**

1. The tutor's profile is successfully created and it is available for students to search and select.

- **Main Success Scenario:**

1. The tutor logs into the system using valid credentials.
2. The tutor goes to the "Create Profile" option.
3. The tutor enters personal details, contact information, and edits the profile picture.
4. The tutor provides qualifications, degree, certifications, and experience in specific subjects.
5. The tutor sets availability time and price for tutoring.
6. The system verifies the details entered by the tutor.

7. After verification, the system publishes the tutor's profile.

- **Extension:**

1. The tutor does not select any subject, and the system warns the tutor to select at least one subject.
2. The tutor does not upload any qualifications or degree.
3. The system fails to verify the tutor's qualifications.

2.1.3 Use Case 3 : Search for Tutor

- **Actor:** Student
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Preconditions:**

1. The student must be logged in.

- **Success Guarantee:**

1. The student gets a list of tutors matching their preferences.

- **Main Success Scenario:**

1. The student logs into the system using valid credentials.
2. The student goes to the "Find Tutor" section.
3. The system displays search filters like subject, availability, rating, and price.
4. The student selects their preferences and searches.
5. The system displays a list of tutors matching the search.
6. The student gets the list and checks tutor profiles.

- **Extension:**

1. The system displays that no tutor matches the search criteria and suggests alternative profiles.

2.1.4 Use Case 4: Book a Tutoring Session

- **Actor:** Student
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Preconditions:**
  1. The student has contacted and selected a tutor.
  2. The tutor has approved the request and is available for the session.
- **Success Guarantee:**
  1. The session is successfully scheduled.
- **Main Success Scenario:**
  1. The student logs into the system using valid credentials.
  2. The student goes to the selected tutor's profile.
  3. The student selects a suitable time slot and submits the session request.
  4. The system sends a notification to the tutor for the session request.
  5. If the tutor accepts the request, the system confirms the booking and notifies the student.
- **Extension:**
  1. The tutor declines the session request.
  2. No time slots are available.

2.1.5 Use Case 5: Attend Tutoring Session

- **Actor:** Student, Tutor
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Preconditions:**
  1. The student has selected a tutor.
  2. The tutor is available for the session.

3. The session is scheduled and confirmed.

- **Success Guarantee:**
  1. Both the student and tutor have successfully joined the video call.
- **Main Success Scenario:**
  1. The student clicks the "Join Session" button in their dashboard.
  2. The tutor also clicks "Join Session" from their dashboard.
  3. The system starts a video call between the student and tutor.
  4. The video session begins, and both can communicate.
- **Extension:**
  1. Poor audio and video quality due to network issues.
  2. No one joined the session.

2.1.6 Use Case 6: Monitor Real-Time Emotional Analytics

- **Actor:** Tutor, Student
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Preconditions:**
  1. The student must have their video camera on.
  2. The emotional analytics feature must be enabled.
- **Success Guarantee:**
  1. The tutor receives real-time emotional feedback during the session.
  2. The tutor can change their teaching strategies according to the emotional feedback.
- **Main Success Scenario:**
  1. When the student and the tutor start the video call, the system begins tracking the student's facial expressions.
  2. The system analyzes the emotions such as confusion, focus, frustration, etc.
  3. The tutor adjusts their teaching strategy according to the feedback.

4. The system provides the emotional feedback for the complete session.
5. At the end, the tutor can review an emotional analytics report.

- **Extension:**

1. The student disables the camera.
2. The system gives incorrect emotions.

2.1.7 Use Case 7: Generate Progress Report

- **Actor:** Tutor, Student, Parent
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Preconditions:**

1. The tutor and student must have attended tutoring sessions.
2. The system must store student performance and emotional analytics data.

- **Success Guarantee:**

1. The tutor receives a detailed report on the student's progress.

- **Main Success Scenario:**

1. The tutor goes to the student's profile and selects the "Generate Report" option.
2. The system analyzes the performance and emotional analytics data.
3. The tutor adds other assessment data and comments.
4. The system provides the data in the form of a report.
5. The tutor reviews the report and sends it to the student and parents.

- **Extension:**

1. The system fails to get the student's data.
2. The system fails to generate the report.

2.1.8 Use Case 8: Provide Feedback

- **Actor:** Student, Parent
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Preconditions:**
  1. The tutor and student must have attended at least one tutoring session.
- **Success Guarantee:**
  1. The feedback is successfully provided and added to the tutor's profile.
- **Main Success Scenario:**
  1. After attending the tutoring session, the student can access the feedback option.
  2. The student rates the tutor and writes a review.
  3. The system checks the review for anything inappropriate.
  4. After the check, the student submits the feedback.
  5. The system updates the tutor's profile by adding the review.
- **Extension:**
  1. The system detects inappropriate content in the review, and the student fails to submit the review.

2.1.9 Use Case 9: Join Social Learning Communities

- **Actor:** Student
- **Scope:** SmartTutor Web Application
- **Level:** User Goal
- **Preconditions:**
  1. The student must be logged in to the platform.
  2. The social learning communities must be available on their account.
- **Success Guarantee:**

1. The user can join, post, and interact in the learning communities.

- **Main Success Scenario:**

1. The student logs into the system using valid credentials.
2. The student goes to the "Join Learning Community" section.
3. The system displays a list of available communities.
4. The student can participate in the community by posting questions and sharing resources.
5. The system checks and notifies the student of successful participation.

- **Extension:**

1. The student is unable to join the community due to age or role restrictions.
2. The student is banned or removed from the community.

2.1.10 Use Case 10: Manage Accounts

- **Actor:** Admin
- **Scope:** SmartTutor Web Application
- **Level:** Admin Function
- **Preconditions:**

1. Admin must be logged in to the platform.

- **Success Guarantee:**

1. Admin can successfully manage the selected accounts, including creating, updating, or deleting.

- **Main Success Scenario:**

1. Admin logs into the system using valid credentials.
2. The admin navigates to the "Manage Accounts" section.
3. The system displays a list of user accounts.
4. Admin selects an account for action.
5. After performing the desired actions, the admin submits the changes.
6. The system updates the changes accordingly.

- **Extension:**

1. The admin tries to delete an account that has pending payments, and the system prevents this action.

2.1.11 Use Case 11: Monitor Platform Activity

- **Actor:** Admin
- **Scope:** SmartTutor Web Application
- **Level:** Admin Function
- **Preconditions:**
  1. Admin must be logged in to the platform.
- **Success Guarantee:**
  1. Admin can successfully monitor the platform activity.
- **Main Success Scenario:**
  1. Admin logs into the system using valid credentials.
  2. The admin goes to the "Monitor Platform Activity" section.
  3. The system displays a list of activities to monitor, such as logins and errors.
  4. Admin selects a specific activity to monitor.

2.1.12 Use Case 12: Manage Payments

- **Actor:** Admin, System
- **Scope:** SmartTutor Web Application
- **Level:** Admin Function
- **Preconditions:**
  1. Admin must be logged in to the platform.
- **Success Guarantee:**
  1. Payments are successfully processed and recorded.
- **Main Success Scenario:**
  1. Admin logs into the system using valid credentials.

2. Admin goes to the “Manage Payments” section.
3. Admin views the pending payments for tutors and billing information for students.
4. The system records the payment actions.
5. Admin reviews the payment history.

- **Extension:**

1. The system experiences any issue during payment processing.

2.1.13 Use Case 13: Send Notification

- **Actor:** System
- **Scope:** SmartTutor Web Application
- **Level:** System Function
- **Preconditions:**

1. Any notification trigger event such as account updates, session reminders, etc., occurs.

- **Success Guarantee:**

1. Notifications are sent by the system to the users based on the event.

- **Main Success Scenario:**

1. Any predefined event occurs.
2. The system generates a notification.
3. The system identifies the users based on the notification type.
4. The system sends the notification to the users.
5. Users receive the notification.

- **Extension:**

1. The system fails to deliver the notification.

2.2 Functional Requirements

In this section, we establish the primary functional requirements for the SmartTutor system. These requirements are focused on the very basics that the system should represent to assist with students, tutors, and parents. Every single one outlines a separate feature that the platform needs in order to provide personalized tutoring experiences, real-time emotional analytics, secure communication or comprehensive progress tracking.

2.2.1 User Interface (UI)

Following are the requirements for module 1:

1. **FR1:** An **easy to use API** system which the tutor and student can quickly be achieved through, React for optimized UI dynamic rendering.
2. **FR2:** The system will be capable of **automatic layout adjustment** based on the screen size (for mobile phones, tablets, laptops) and shall work in a responsive way to provide an optimal user experience across platforms.

2.2.2 User Management

Following are the requirements for module 2:

1. **FR1:** The System will provide an admin **Dashboard for managing User** account and User Profile like Add, Update, Delete and View students/Parent/Tutor profile.
2. **FR2:** The system should **provide the admin with the visibility** to follow what users are doing like log in history, session data, how they are using platform and that they comply to proper usage of defined rules of the platform.

2.2.3 Authentication and Authorization

Following are the requirements for module 3:

1. **FR1: Authentication shall be secured** to create and authenticate the users with encrypted credentials, in order to avoid unauthorized access of user data.
2. **FR2:** The system will have fine-grained **role-based access control** to grant a minimum functionality for different kinds of users, who are students, tutors, parents and admin.

2.2.4 Security and Privacy

Following are the requirements for module 4:

1. **FR1: End-to-end encryption** for video calls between tutors and students to secure communication.
2. **FR2:** All user **data will be encrypted**, both in transit and at rest, so a third party cannot access sensitive information like personal details or login credentials.

2.2.5 Version Control and Data Backup

Following are the requirements for module 5:

1. **FR1:** The system shall have **automatic regular data backup** so that user data (account info, session logs) is maintained and can be recovered should it ever be lost.
2. **FR2: Data redundancy** to store another copy of important data in more than one place, ensuring that critical systems can be restored in the event of a disk or system failure.

2.2.6 Real-time Synchronization

Following are the requirements for module 6:

1. **FR1:** The system has to ensure that there is **no delay in the transmission** of audio and video streams between tutors and students during video calling sessions, so it provides real-time synchronization.
2. **FR2:** The system will provide **real-time capture and transmission** of Emotional Data when video-calling so as to provide immediate feedback to the tutors.

2.2.7 Tutor Matching Based on Student Preferences

Following are the requirements for module 7:

1. **FR1:** Students will have the ability to **search for tutors** that fit with their learning style based on subjects, availability and language allowing the student a more personalised tutor matching their learning requirements.

2. **FR2:** The system will show **extensive tutor profiles** including qualifications, experience and subject matter as to allow the students so as to choose or select a proper tutor.

2.2.8 Video Calling

Following are the requirements for module 8:

1. **FR1:** There will be a provision for **secure video and audio streaming**, to ensure lag-free communication of students with the tutors during tutoring sessions.

2.2.9 Real-time Emotional Analytics

Following are the requirements for module 9:

1. **FR1:** The system shall use **emotion detection** to recognise the emotional state (satisfaction, confusion, frustration) of a student during video calls.
2. **FR2:** The system will give **real-time emotional feedback** to tutors and this data can be used by the tutor to modify their teaching strategies and engage students better in class sessions.

2.2.10 Social Learning Communities

Following are the requirements for module 10:

1. **FR1:** The system shall **enable students to interact** with each other and tutors within the community spaces through joining of groups for collaborative learning.

2.2.11 Report Generation

Following are the requirements for module 11:

1. **FR1: Progress reports** that summarize student learning outcomes, session attendance, and emotional engagement during tutoring sessions will be generated for parents by the system.
2. **FR2:** The system will additionally **enable parents to download the reports** generated with a PDF/PNG exportable format to have an easy access and share of their child's progression details.

2.2.12 Feedback

Following are the requirements for module 12:

1. **FR1:** The system would give the user a feedback form to put in the **suggestions and feedback** of the parent and student on tutors so that everybody could share their thoughts without giving own identity.

2.3 Non-Functional Requirements

This is part of our format for an outline that specifies non-functional requirements for the SmartTutor platform. These quality properties are essential to the functioning of a system and its usefulness. With a view to driving designate development and evaluation of the proposed SmartTutor platform, herein we specify the core non-functional requirements.

2.3.1 Reliability

Reliability indicates the level of consistency that you expect from SmartTutor to be able to perform the service. These are the requirements that specify reliability aspects of the system:

1. **Mean Time Between Failures (MTBF):** The system will have a Mean Time Between Failure (MTBF) of at least 1,000 hours, having low failures times is important to providing an efficient learning experience for the students and tutors.

2. **Definition of failure:** Any event that causes the platform to fail to perform a centrally critical ability (for example, video calling operation, user authentication operation or database content retrieval) is called as software failure.

3. **Consequences of Failure:** Software failed and tutoring sessions were broken, critical data is lost, end users lose trust. The system will include strong error handling and notification mechanisms to let the user know if there are potential issues as soon as possible.

4. **Protection from Failure:** It will also implement strategies like input validation, regular code reviews and automated testing to help prevent failures. All ingredients in this list must pass stress tests and be capable of handling maximum load.

5. **Error Detection Strategy:** The system will employ logging and monitoring mechanisms to enable real-time detection of errors. Alerts will be generated to admins

in case of catastrophic failure or performance degradation, hence implying quick identification and response.

6. **Error Correction Strategy:** The system shall be equipped with roll-back capability in case it fails so as to restore the previous stable state. The system shall also have a dedicated support team whose job will primarily entail addressing the issues and determining remedies to ensure swift corrective actions to get things back on the normal track.

2.3.2 Usability

Usability requirements focus on ensuring that the SmartTutor platform provides an intuitive and efficient user experience. These requirements encompass ease of learning, ease of use, error avoidance and recovery, the efficiency of interactions, and accessibility. The specified usability requirements will assist the user interface designer in creating an optimum user experience.

1. **USE-1:** The system should allow users to register and login within three interactions at most, in addition to keeping the sign-up and authentication process as straightforward as possible.

2. **USE-2:** Help on the system should be available from every page so immediate assistance and resources to use the system properly will be given.

3. **USE-3:** The system shall have a clear navigation, meaning it shall be easy for a user to find access to one of the functions, such as tutor matching and video calling, within two clicks from the dashboard.

4. **USE-4:** The platform should use error messages that are friendly to the user and also indicate how such errors may be rectified in case of input error at the time of registration, login, or tutor selection processes.

5. **USE-5:** The system shall embrace accessibility standards, which should ensure that users with facing difficulties can access and employ the platform effectively.

6. **USE-6:** The platform should enable users to provide information relating to their experience with the session undertaken with each tutor to facilitate continuous improvement in usability as informed by user feedback.

7. **USE-7:** The system should enable users to update their profiles with personal information and preferences in at least three interactions to ensure the process can be as smooth and efficient as possible.

8. **USE-8:** The platform will be providing visible feedback in the case of successful actions like form submission, attempted joining a video call, and saved settings-for example, through loading icons or confirmation messages-to remind users that their actions are successfully being processed.

2.3.3 Performance

This section describes the performance requirements for most operations of the system. The requirements ensure that the SmartTutor application operates efficiently and effectively under expected usage conditions.

1. **PER-1:** The system will ensure that 95 percent of video calls set between students and tutors are connected within 3 seconds with an Internet connection of at least 20 Mbps or higher.
2. **PER-2:** The platform will process and return the results of tutor searches based on student preferences within 2 seconds for 90 percent of queries.
3. **PER-3:** The system will be able to support as many as 500 simultaneous users without degrading performance. As such, all users can still enjoy smooth interactions even during peak usage times.
4. **PER-4:** The site shall generate and display a progress report to parents within 5 seconds of the request being placed. Access to critical information will thus be prompt.
5. **PER-5:** The system shall provide the user with the facility of upload and sharing of resources with maximum upload time being 5 seconds for file sizes of up to 10 MB over a standard broadband connection.
6. **PER-6:** The system shall be able to provide feedback on real-time emotional analytics regarding changes in the emotional state of the student to the tutors within 5 seconds during video calls to adjust teaching strategies well in time.
7. **PER-7:** The platform shall have the load time of the user interface for users within 2 seconds of getting to the homepage or the dashboard at 90 percent.

2.3.4 Security

In this section, security requirements that protect data and the SmartTutor platform were specified. These security requirements aim to prevent unauthorized access, support data breach protection, and enforce data confidentiality, integrity, and availability for the user.

1. **SEC-1:** The system should be capable of withstanding, without any breach, at least 99.9 percent attempted accesses to it in an unauthorized manner.

2. **SEC-2:** The platform shall ensure that sensitive information related to its users remains protected- both personal and academic information- such that during any possible breach, no more than 0.01 percent of the information is leaked.

3. **SEC-3:** The platform shall detect and log all failed login attempts and suspicious activity; such can be reviewed and analyzed within 1 hour of the event so as not to expose it for any longer period.

4. **SEC-4**: The system will protect 100 percent of all user data, including communication between students, tutors, and parents through encryption protocols while in transit so that it is impossible to intercept or tamper with such data.

Chapter 3

System Overview [AS PER FYP1 MID REPORT]

Give a general description of the functionality, context, and design of your project. Provide any background information if necessary.

3.1 Architectural Design

The architecture provided represents a three-tier architecture. Here's a breakdown of each layer:

1. **Presentation Layer**
    - **Description:** This is the layer where the user's interface and his interaction are given. Here, all the functionalities involved with the display of information along with taking user input are handled.
    - **Components:**
        - **Web Interface:** This allows users to interact with the system via a GUI.
2. **Application Layer**
    - **Description:** It is also referred to as the business logic layer. This is the layer where the core application functionalities reside. It processes user inputs, executes operations based on predefined business rules, and communicates with the data layer.
    - **Components:**
        - **User Management Module:** Handles user registration, login, and profile management.

- **Session Management Module:** Manages tutoring sessions, including scheduling and attendance.
- **Analytics Module:** Processes data related to user performance and emotional analytics.
- **Progress Module:** Tracks student progress and generates reports.
- **Feedback Module:** Collects and manages feedback from users.
- **Community Module:** Facilitates social interactions and community features.

3. **Data Layer**

   - **Description:** This layer is responsible for data storage and management. It interacts with the database to perform operations like create, read, update, and delete (CRUD).
   - **Components:**
     - **Database:** The relational database (like MySQL or PostgreSQL) where all data entities (users, sessions, feedback, etc.) are stored.

Conclusion: The architecture follows a three-tier model, where:

- The Presentation Layer is concerned with user interface and experience.
- The Application Layer focuses on business logic and processes.
- The Data Layer is responsible for data management and storage.

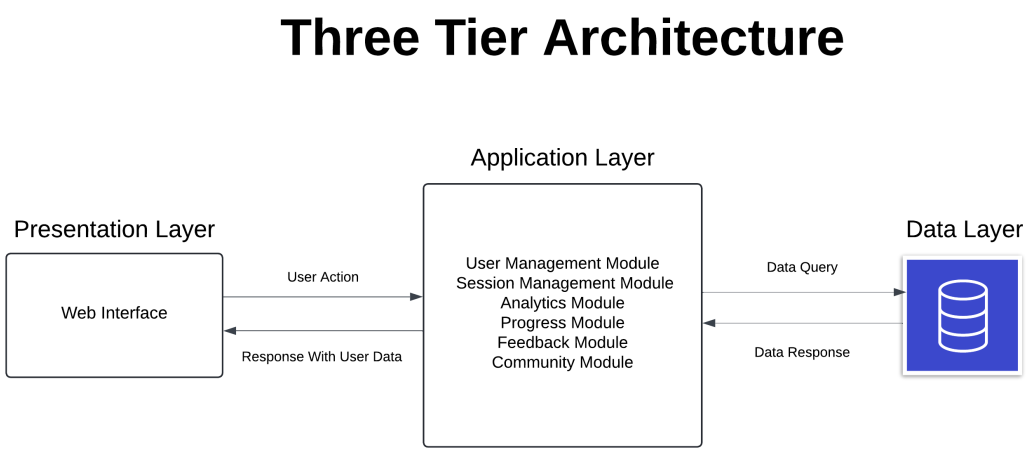

Figure 3.1: Architecture Diagram

3.2 Data Design

In the SmartTutor system, the information domain is transformed into various data structures for the purpose of efficient storage, processing, and further organization of large-sized data entities: users (students, tutors, parents, admins), sessions, feedback, emotional analytics, preferences, and payments.

3.2.1 Major Data Entities

Here are the primary data entities involved in the SmartTutor system, detailing their attributes:

3.2.1.1 Users

- **Table Name:** Users - **Attributes:** - `UserID` (Primary Key): Unique identifier for each user. - `Username`: Unique username for user login. - `Password`: Hashed password for security. - `Email`: Email address of the user. - `Role`: Specifies the user's role (Student, Tutor, Parent, Admin). - `RegistrationDate`: Date when the user registered. - `PhoneNumber`: Contact number of the user.

3.2.1.2 TutorProfiles

- **Table Name:** `TutorProfiles` - **Attributes:** - `TutorID` (Primary Key, Foreign Key referencing UserID): Unique identifier for the tutor. - `Qualifications`: Educational qualifications of the tutor. - `SubjectsSpecialization`: Subjects the tutor specializes in. - `Rating`: Average rating based on feedback. - `AvailabilitySchedule`: Tutor's available hours for sessions. - `Experience`: Number of years of tutoring experience.

3.2.1.3 StudentProfiles

- **Table Name:** `StudentProfiles` - **Attributes:** - `StudentID` (Primary Key, Foreign Key referencing UserID): Unique identifier for the student. - `Grade`: Current grade level of the student. - `Interests`: Subjects or activities the student is interested in. - `CurrentTutor`: ID of the current tutor assigned to the student. - `GuardianName`: Name of the guardian or parent. - `School`: School the student attends. - `Preferences`: Preferences related to learning styles or subjects.

3.2.1.4 ParentProfiles

- **Table Name:** `ParentProfiles` - **Attributes:** - `ParentID` (Primary Key, Foreign Key referencing UserID): Unique identifier for the parent. - `Children`: List of children associated with this parent. - `ContactInformation`: Contact details for the parent.

3.2.1.5 AdminProfiles

- **Table Name:** `AdminProfiles` - **Attributes:** - `AdminID` (Primary Key, Foreign Key referencing UserID): Unique identifier for the admin. - `Permissions`: Specific permissions granted to the admin. - `Role`: Defines the level of admin access.

3.2.1.6 Sessions

- **Table Name:** `Sessions` - **Attributes:** - `SessionID` (Primary Key): Unique identifier for each tutoring session. - `StartTime`: Start time of the session. - `EndTime`: End time of the session. - `Duration`: Duration of the tutoring session (in minutes). - `Status`: Current status of the session (Scheduled, Completed, Cancelled). - `VideoLink`: Link to the video meeting (if applicable). - `EmotionalAnalytics`: Any emotional analytics associated with the session.

3.2.1.7 Feedback

- **Table Name:** `Feedback` - **Attributes:** - `FeedbackID` (Primary Key): Unique identifier for each feedback entry. - `FeedbackText`: Text comments provided by the student regarding the session. - `Rating`: Rating given by the student (e.g., on a scale of 1-5). - `IsAnonymous`: Indicates whether the feedback is anonymous. - `DateSubmitted`: Date when the feedback was submitted.

3.2.1.8 Reports

- **Table Name:** Reports - **Attributes:** - `ReportID` (Primary Key): Unique identifier for each report. - `StudentID` (Foreign Key referencing UserID): The ID of the student for whom the report is generated. - `TutorID` (Foreign Key referencing UserID): The ID of the tutor involved in the sessions. - `EmotionalAnalyticsSummary`: Summary of emotional analytics for the student. - `PerformanceScore`: Score or data representing the student's performance. - `GeneratedDate`: Date when the report was generated.

3.2.1.9 EmotionalAnalytics

- **Table Name:** `EmotionalAnalytics` - **Attributes:** - `EmotionID` (Primary Key): Unique identifier for emotional analytics entry. - `SessionID` (Foreign Key referencing SessionID): The ID of the session being analyzed. - `StudentID` (Foreign Key referencing UserID): The ID of the student whose emotions are being analyzed. - `EmotionalType`: Type of emotion (e.g., Happy, Frustrated). - `Timestamp`: Date and time of the emotional state recording.

3.2.1.10 Payments

- **Table Name:** `Payments` - **Attributes:** - `PaymentID` (Primary Key): Unique identifier for each payment. - `StudentID` (Foreign Key referencing UserID): The ID of the student making the payment. - `TutorID` (Foreign Key referencing UserID): The ID of the tutor receiving the payment. - `Amount`: Total amount paid. - `PaymentDate`: Date of the payment transaction. - `PaymentMethod`: Method used for payment (e.g., Credit Card, PayPal). - `PaymentStatus`: Status of the payment (e.g., Completed, Pending).

3.2.1.11 Preferences

- **Table Name:** `Preferences` - **Attributes:** - `PreferenceID` (Primary Key): Unique identifier for each preference entry. - `UserID` (Foreign Key referencing UserID): The ID of the user associated with the preferences. - `PreferredTutor`: ID of the preferred tutor for the user. - `LearningStyle`: Preferred learning style (e.g., Visual, Auditory). - `SubjectPreferences`: Subjects the user prefers for learning.

3.2.2 Data Storage and Organization

The data for the SmartTutor system is organized into structured tables within a relational database management system (RDBMS) such as MySQL. The database schema is designed to support the following tables and relationships:

1. **Users Table:** Stores basic user information that is common to all users (students, tutors, parents, and admins).
2. **TutorProfiles Table:** Stores additional profile details specific to tutors, with a foreign key relationship to the Users table.
3. **StudentProfiles Table:** Stores additional profile details specific to students, with a foreign key relationship to the Users table.
4. **ParentProfiles Table:** Stores information specific to parents, with a foreign key relationship to the Users table.

5. **AdminProfiles Table:** Contains information specific to admins, with a foreign key relationship to the Users table.
6. **Sessions Table:** Logs all scheduled tutoring sessions, linking students and tutors through foreign keys.
7. **Feedback Table:** Captures feedback from students after sessions, linking back to the Sessions table through a foreign key.// 8. **Reports Table:** Maintains performance reports for users, linking back to the Users table through a foreign key.
9. **EmotionalAnalytics Table:** Stores emotional analytics data linked to each session and user.
10. **Payments Table:** Logs payment transactions associated with sessions, linking students and tutors.
11. **Preferences Table:** Stores user preferences related to learning styles and subjects.

3.2.3 Data Processing

The SmartTutor system performs several operations in data processing to ensure efficient management and manipulation of data. Key functionalities through processing are as follows:

- **CRUD Operations:** Each entity is permitted to support all or some CRUD operations, extending it to manage user accounts, session bookings, feedback, emotional analytics, preferences, and payment transactions.
- **Data Validation:** Input validation ensures that inputted data is consistent and integrity-preserving, such as ensuring valid email formats, enforcing user roles, and validating payment amounts.
- **Querying:** SQL queries are used to fetch data from the database, including available tutors, retrieving session histories, generating performance reports, and tracking payments.

3.2.4 Data Security

To ensure data security, the system employs several measures:

- **Encryption:** User passwords are stored in a hashed format using secure hashing algorithms, and payment information is also encrypted.
- **Access Control:** Role-based access control is implemented to restrict access to sensitive data based on user roles.

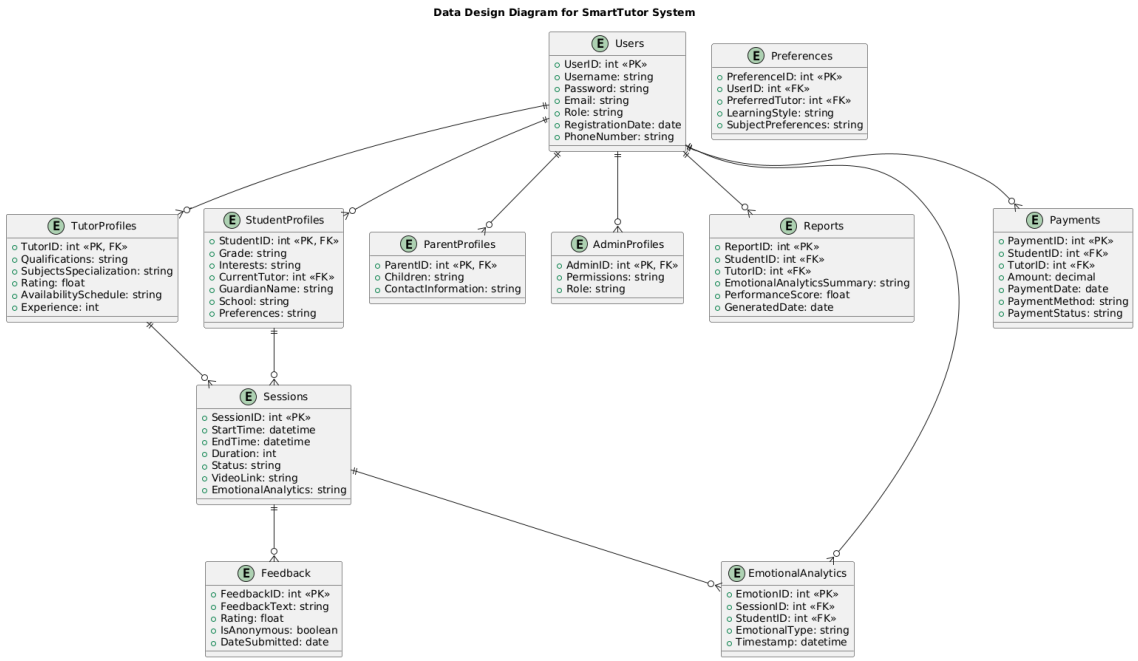

Figure 3.2: Data Design Diagram

3.3 Domain Model

Following is the domain model which provides a visual representation of the key entities within the SmartTutor platform and their relationships.

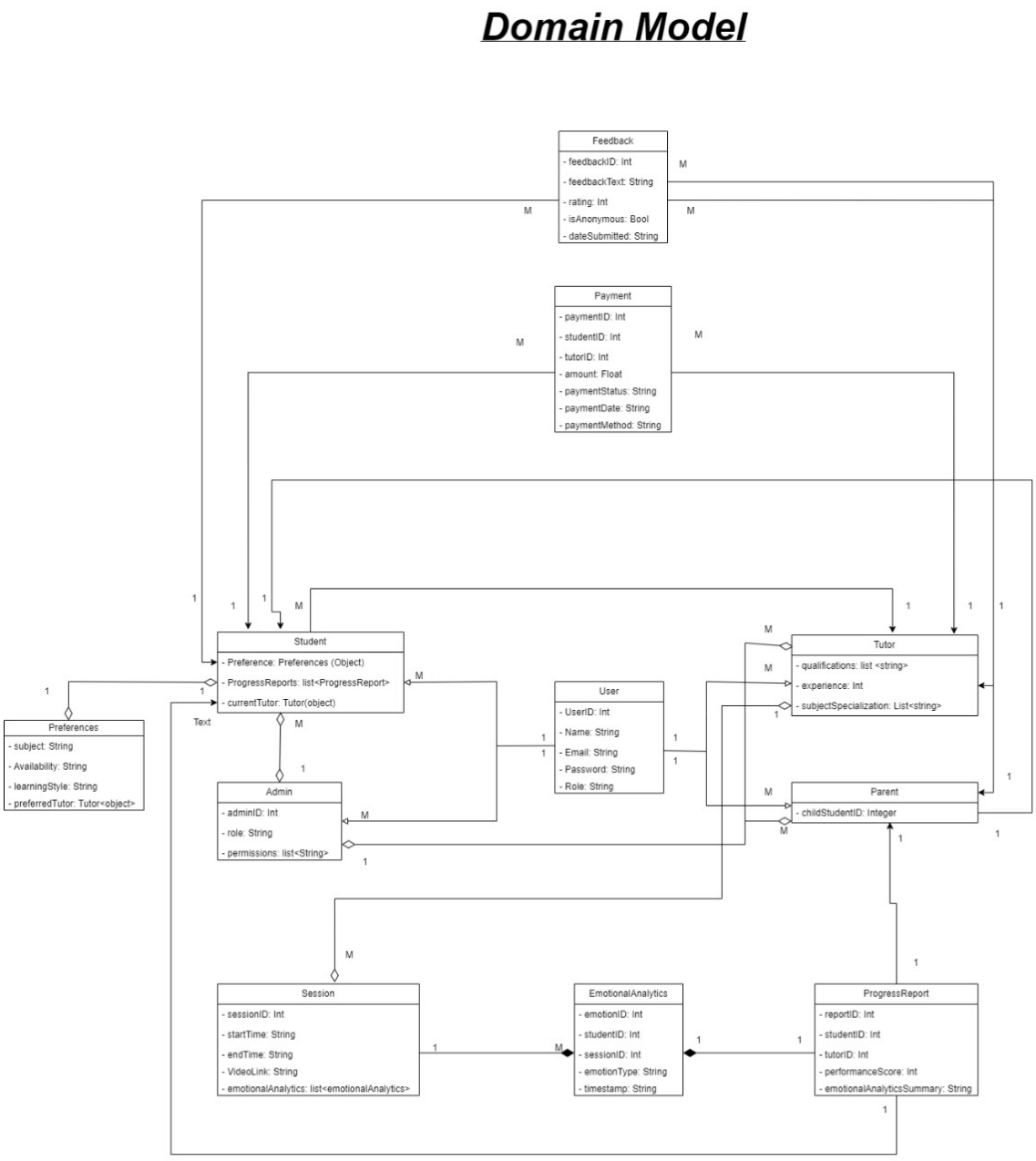

Figure 3.3: Domain Model

3.4 Design Models

This part describes the design models that can be applied to the SmartTutor system, working with an object-oriented development. Every model contains deep descriptions together with visual representations to better support the architecture and interaction of the system. The models used for describing this system are:

• Activity Diagram

• Class Diagram
• Class-level Sequence Diagram
• State Transition Diagram

These diagrams illustrate both dynamic behavior and structural organization of the system thus achieving clearness and transparency in the whole phase of design.

Design Models for Object Oriented Development Approach

3.4.1 Activity Diagram

The activity diagram is used to represent the flow of activities and actions in the Smart-Tutor system. It depicts the processes involved and their sequential order; hence, it demonstrates interaction between users and the system. In this respect, it represents how different activities are interconnected so that the workflow of the application can be understood—user actions, decision points, and parallel processes. This model is basic to the assessment of operation efficiency and the finding of areas in user interaction and system functionality that need to be optimized.

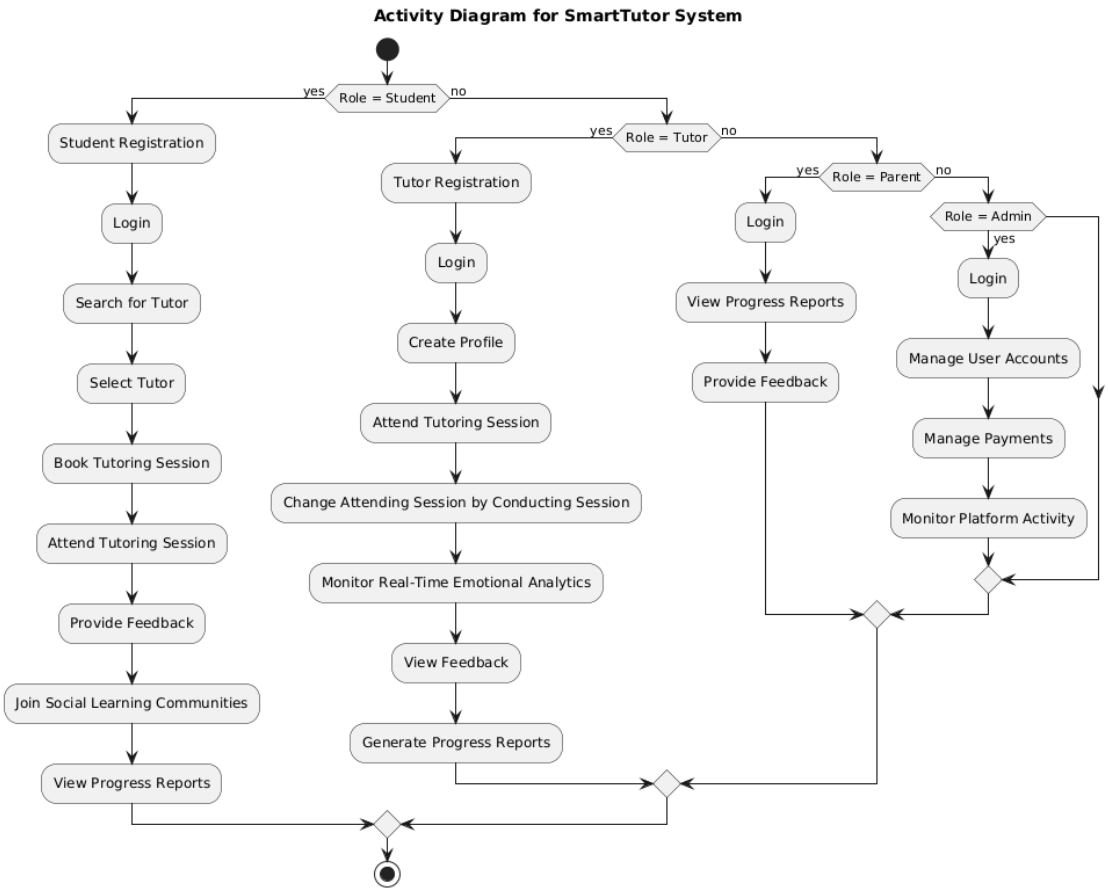

Figure 3.4: Activity Diagram

3.4.2 Class Diagram

The class diagram is a basic view of SmartTutor, mapping the structure of the system together with associated relationships between its fundamental entities. All classes in the system are indicated along with their attributes, methods, as well as visibility. Being able to depict different classes interact with the use of associations, inheritance, and dependencies, makes the class diagram very suitable to give an ideal view of object-oriented design. This is the model that would allow people to understand how an application was structured and thus talk about the application coherently for the development process with stakeholders.

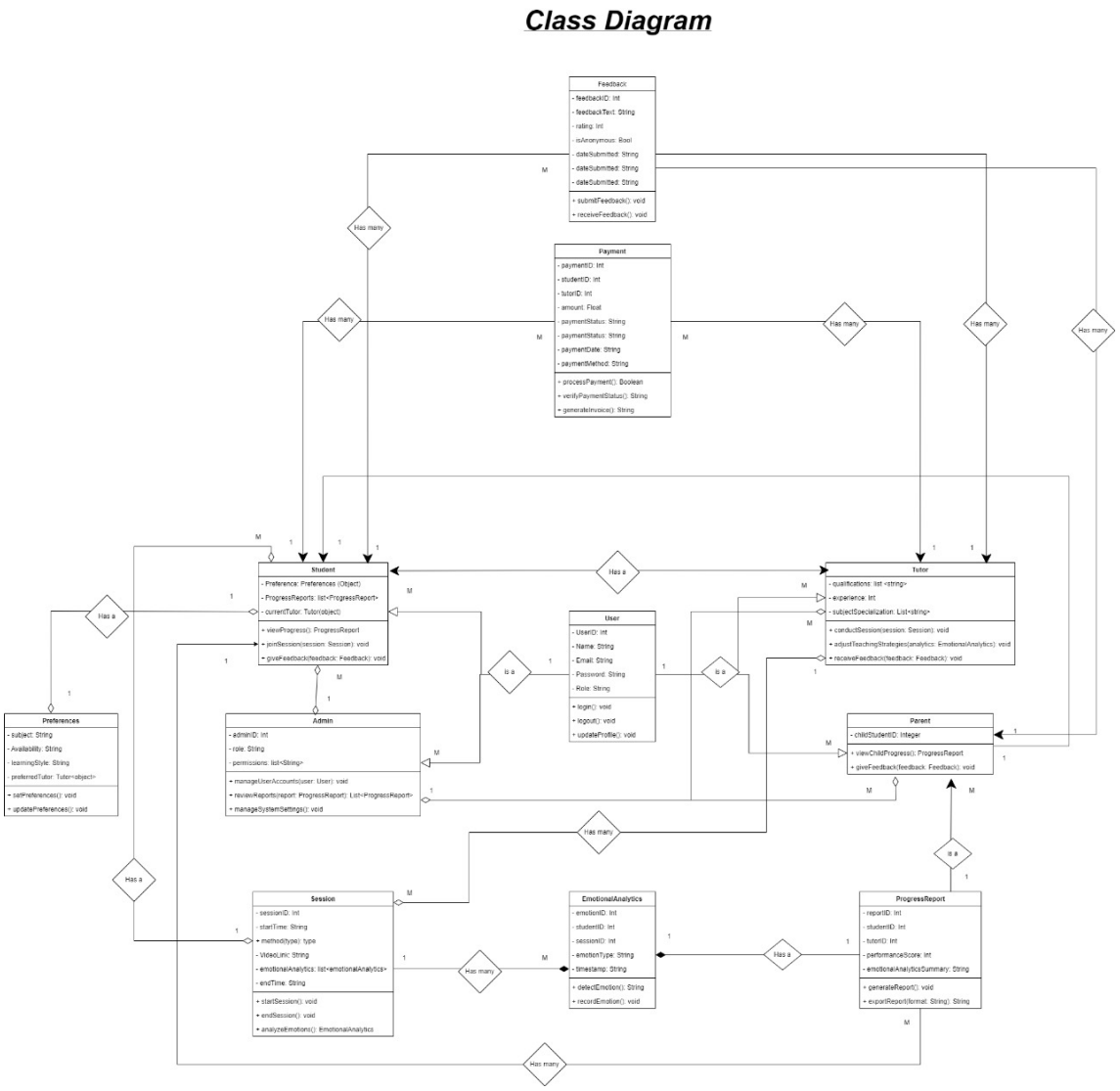

Figure 3.5: Class Diagram

3.4.3 Sequence Diagram

The sequence diagram is a very essential tool used to demonstrate the time-evolving behavior of the objects within the SmartTutor. It represents the communication or message sequence among the classes and different components in a specific use case or scenario. It reveals the control flow and timing aspect involved in achieving some particular task through the way in which objects collaborate. The model has been very helpful in understanding exactly how the application works, identifying those potential bottlenecks, and making sure all interactions are aligned with overall system requirements.

3.4.3.1 SSD 1: Student Register

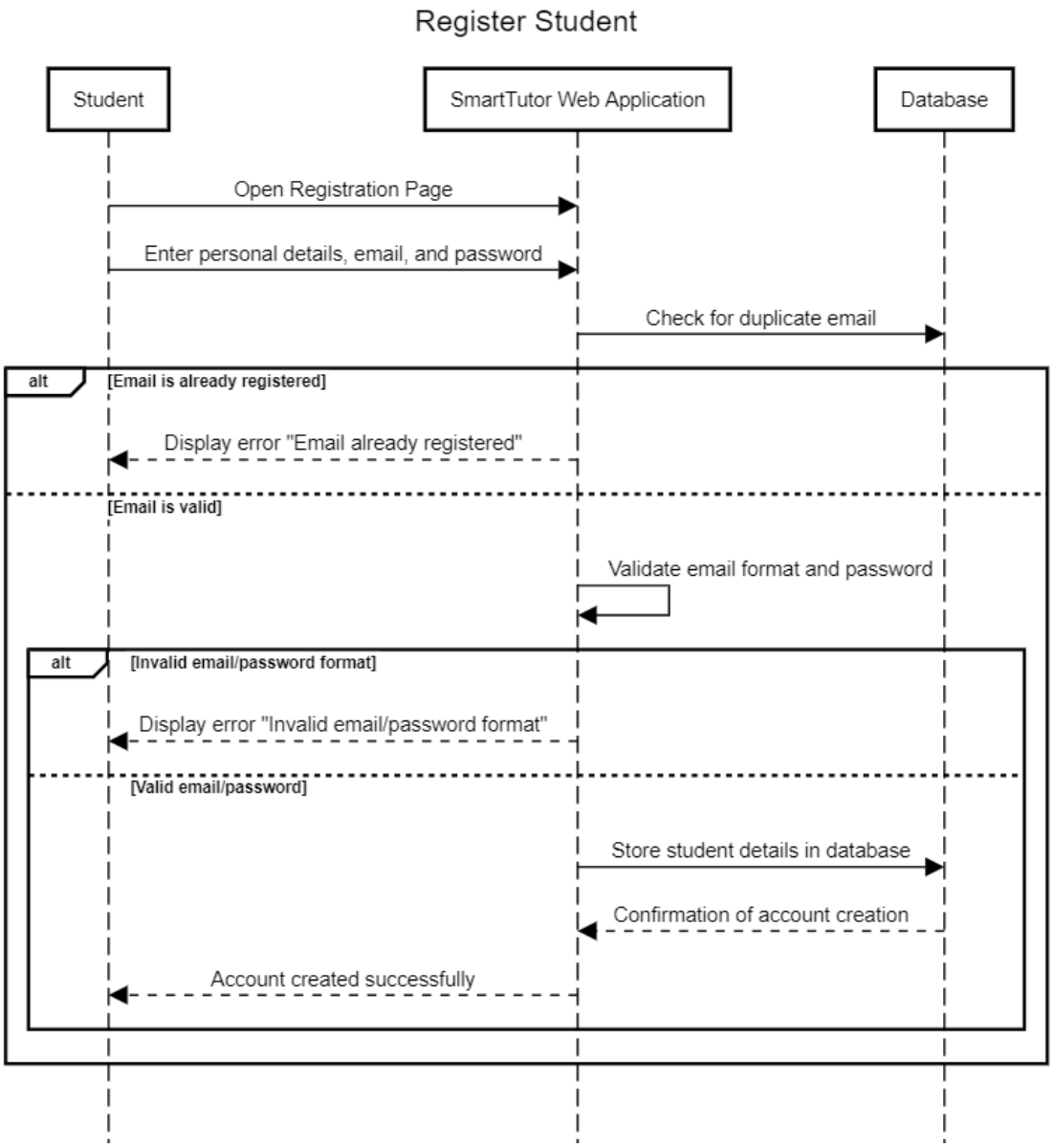

Figure 3.6: SSD (Student Register)

3.4.3.2 SSD 2: Tutor Profile Creation

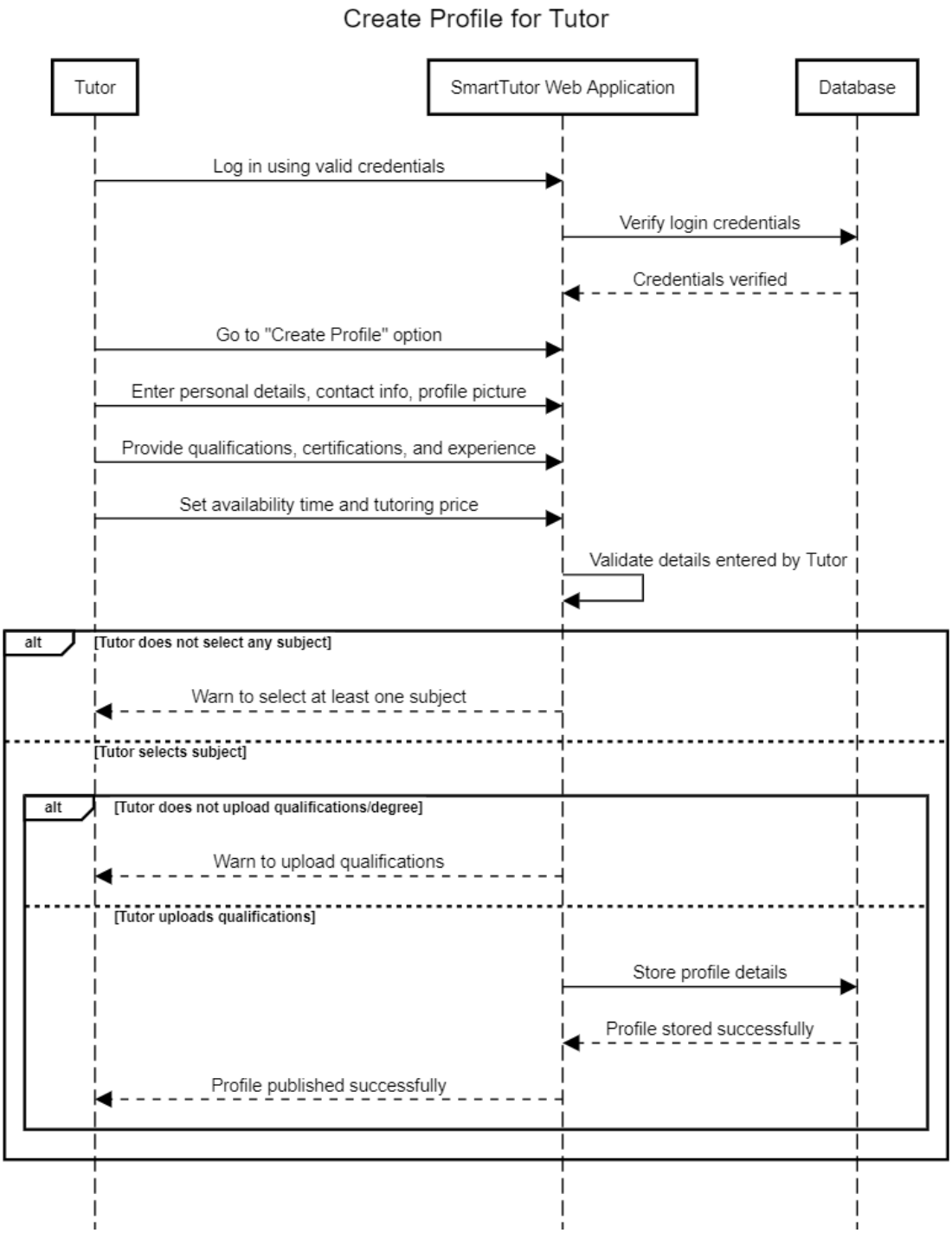

Figure 3.7: SSD (Tutor Profile Creation)

3.4.3.3 SSD 3: Search For Tutor

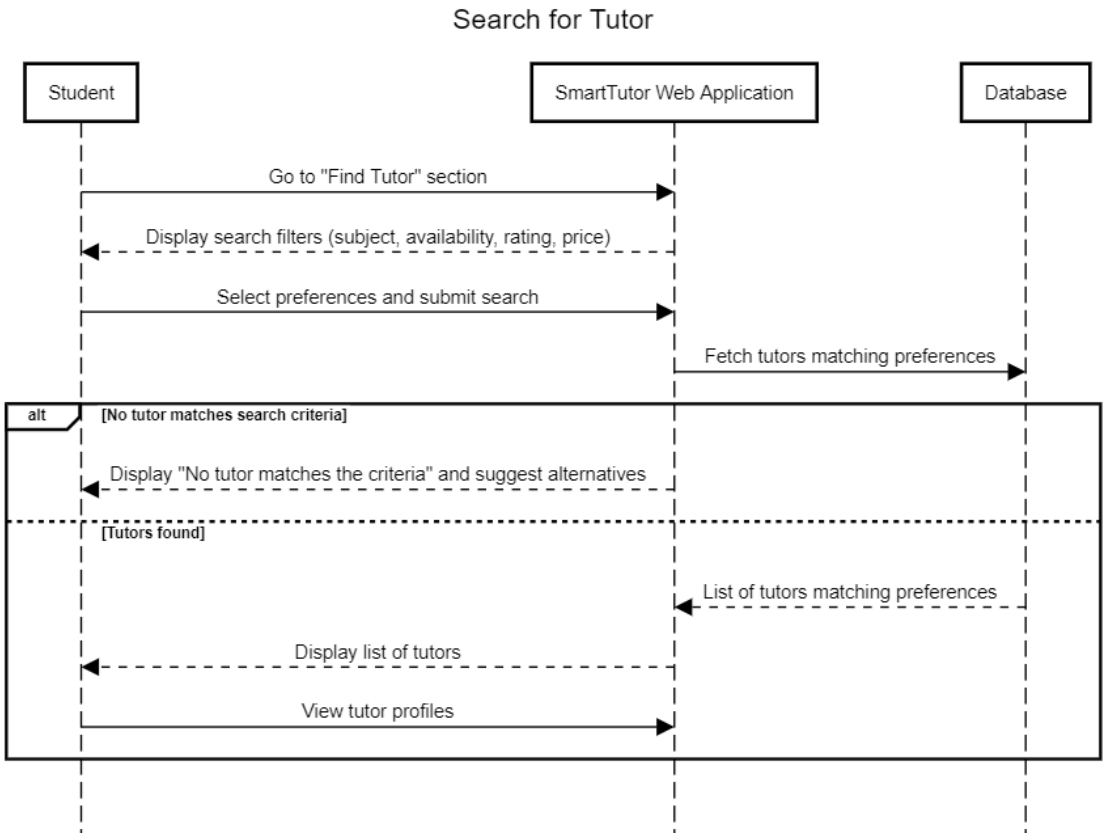

Figure 3.8: SSD (Search For Tutor)

3.4.3.4 SSD 4: Book Tutoring session

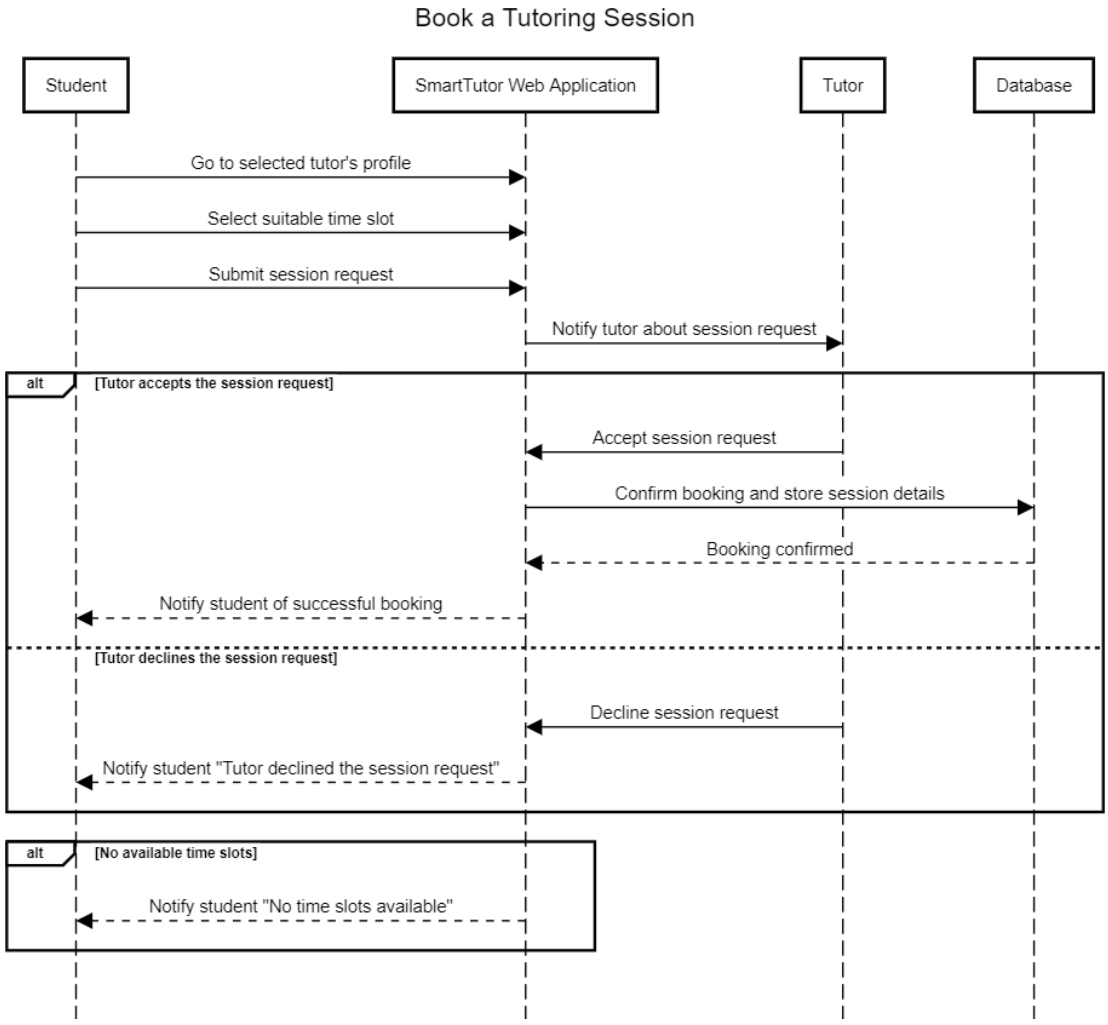

Figure 3.9: SSD (Book Tutoring session)

3.4.3.5 SSD 5: Attend Session

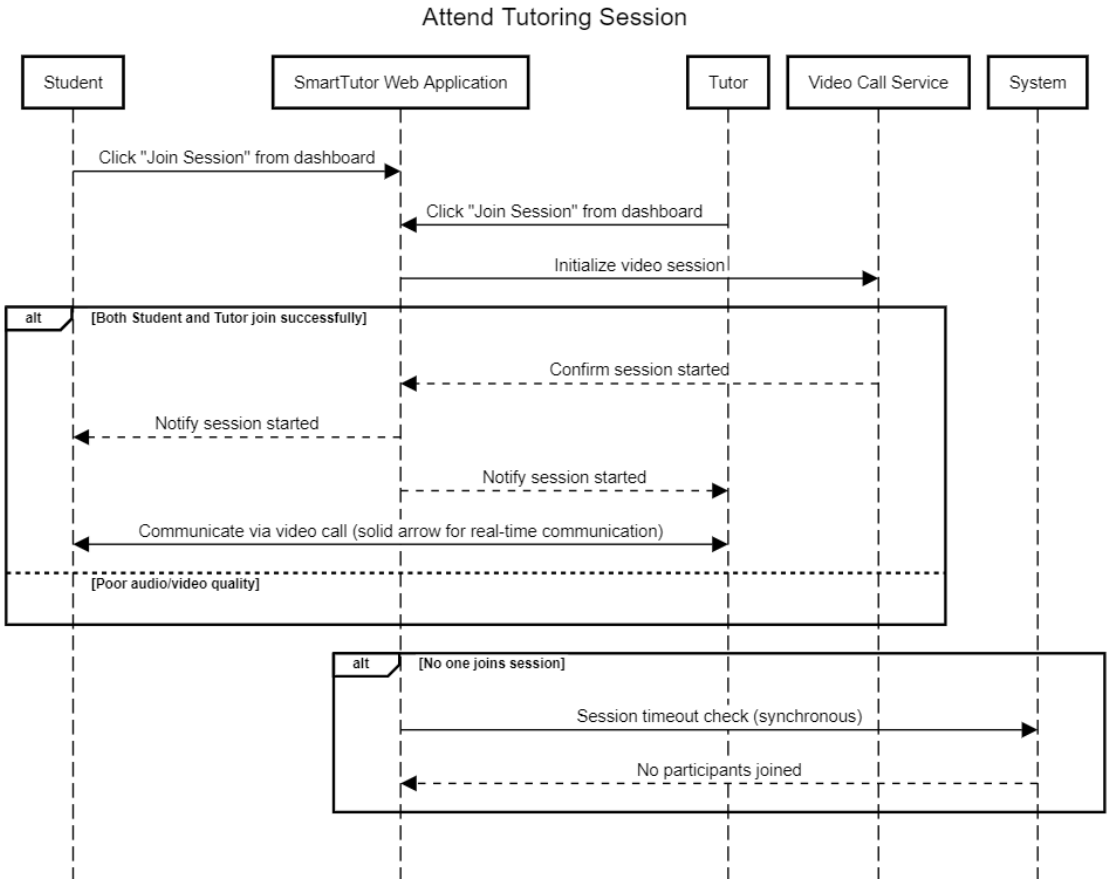

Figure 3.10: SSD (Attend Session)

3.4.3.6 SSD 6: Emotional Analytics

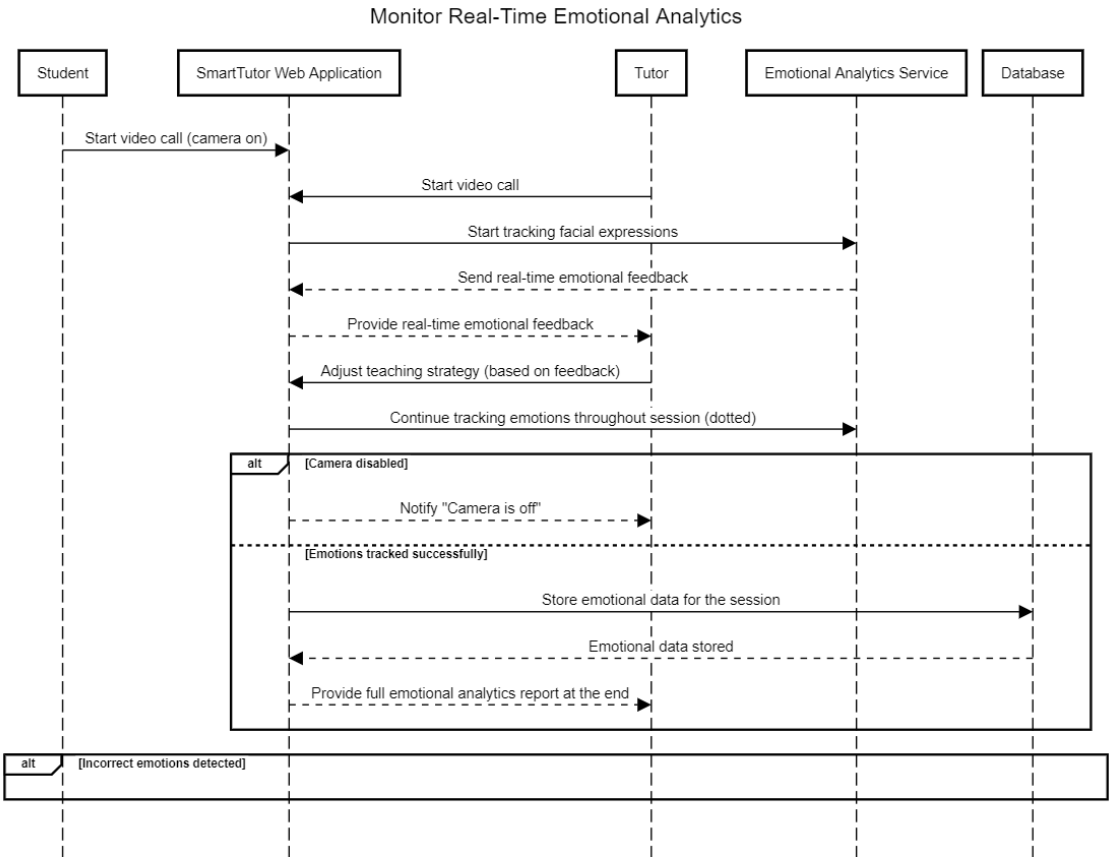

Figure 3.11: SSD (Emotional Analytics)

3.4.3.7 SSD 7: Generate Progress Report

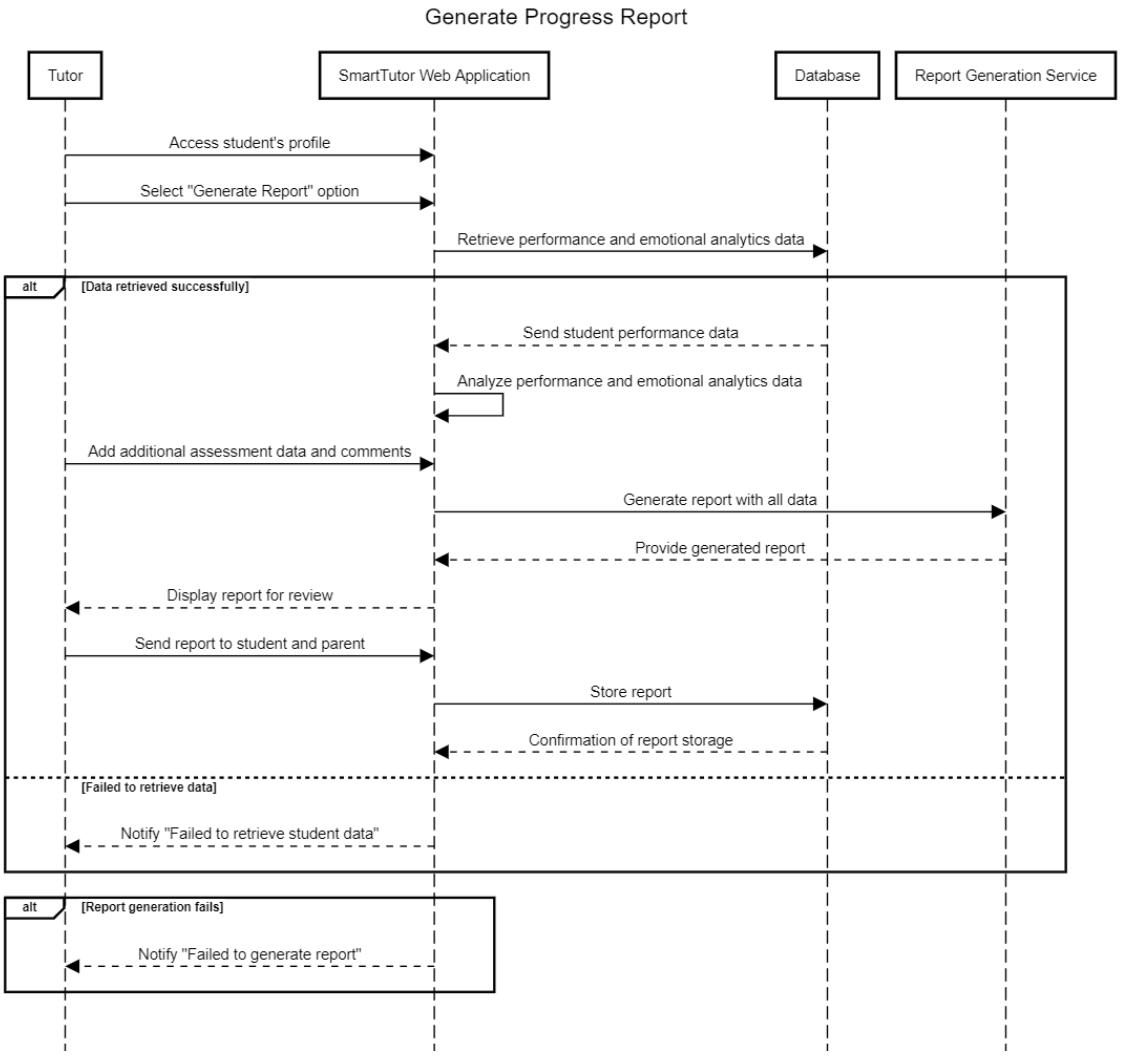

Figure 3.12: SSD (Generate Progress Report)

3.4.3.8 SSD 8: Feedback

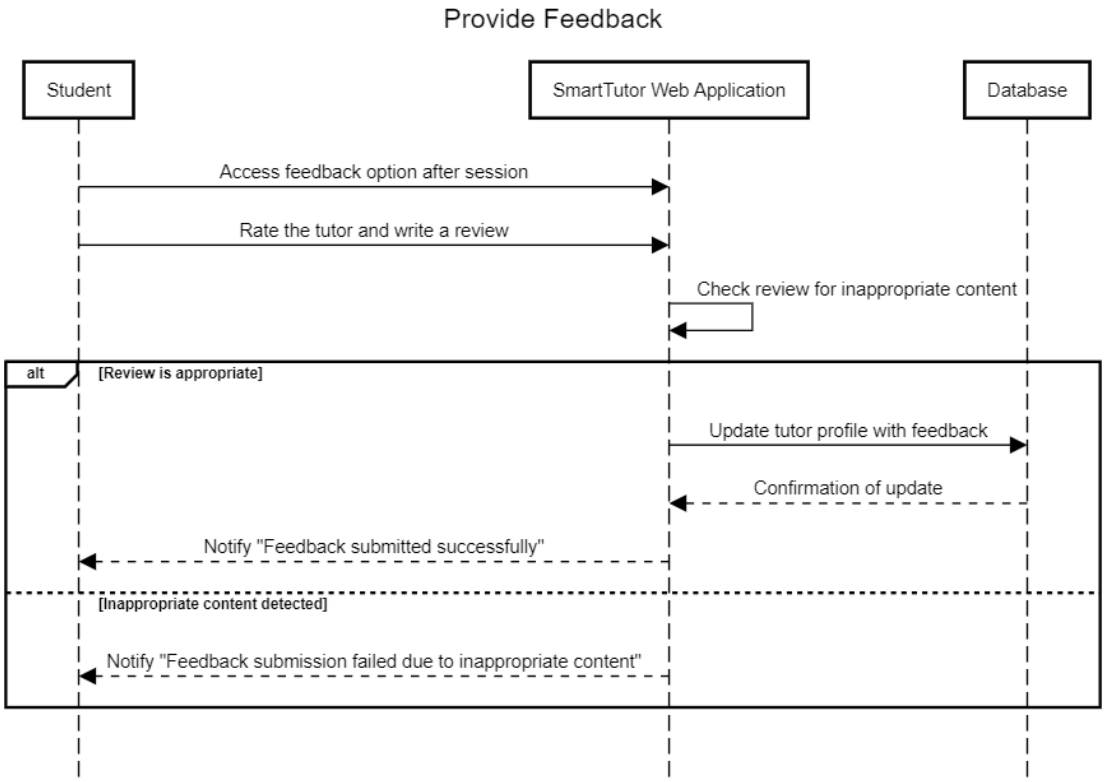

Figure 3.13: SSD (Feedback)

3.4.3.9 SSD 9: Join Communities

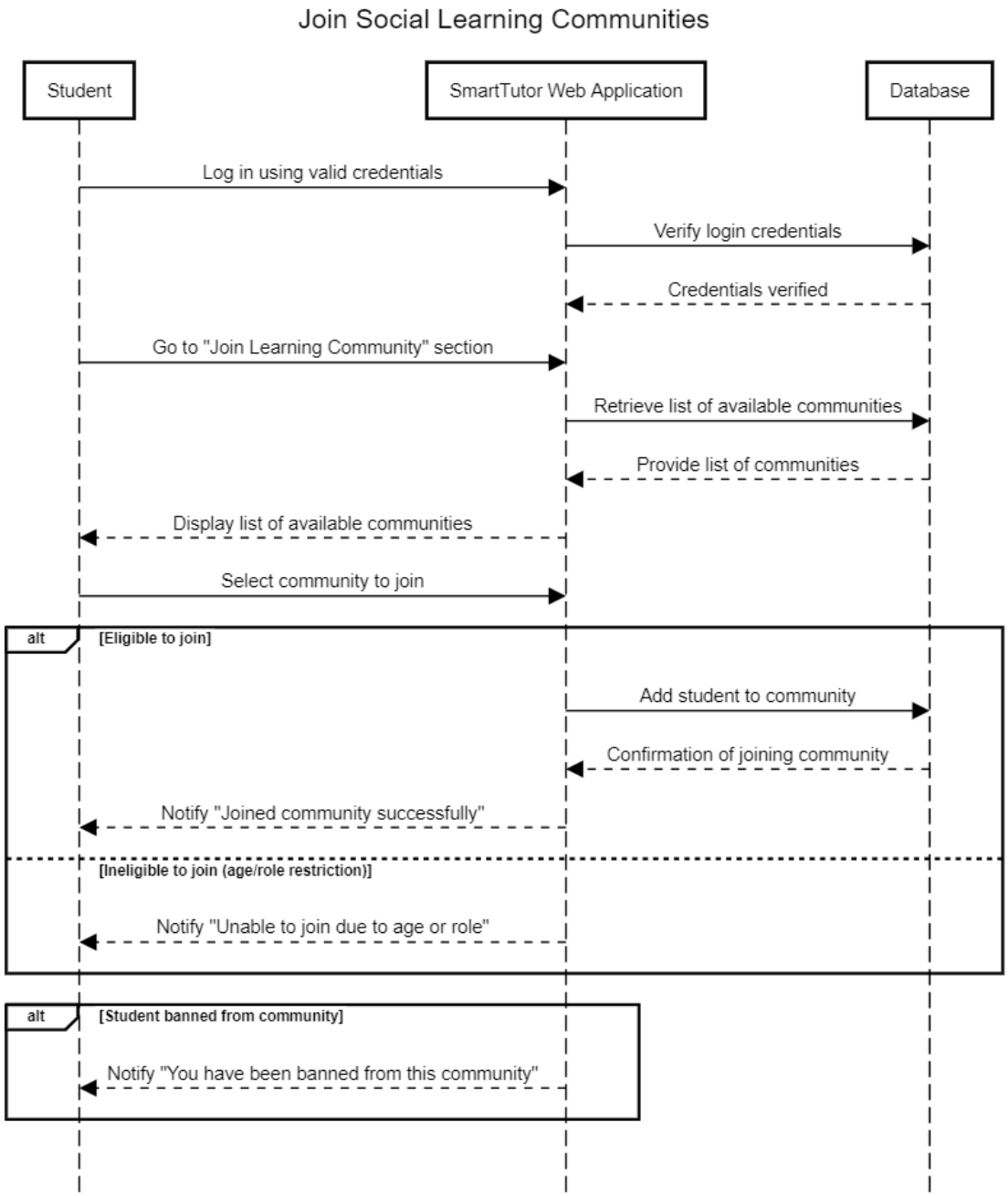

Figure 3.14: SSD (Join Communities)

3.4.3.10 SSD 10: Manage Account

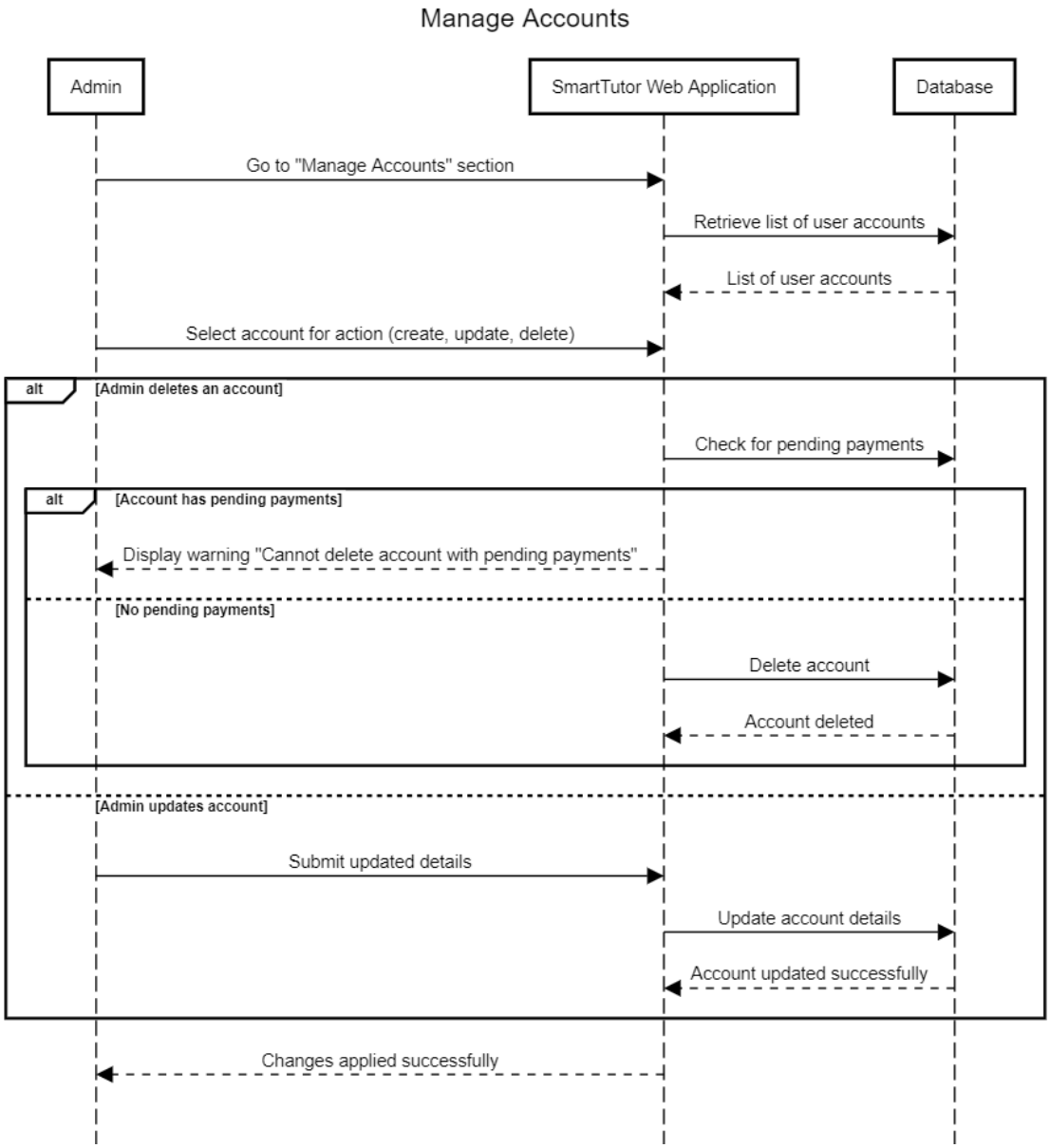

Figure 3.15: SSD (Manage Account)

3.4.3.11 SSD 11: Monitor Platform Activity

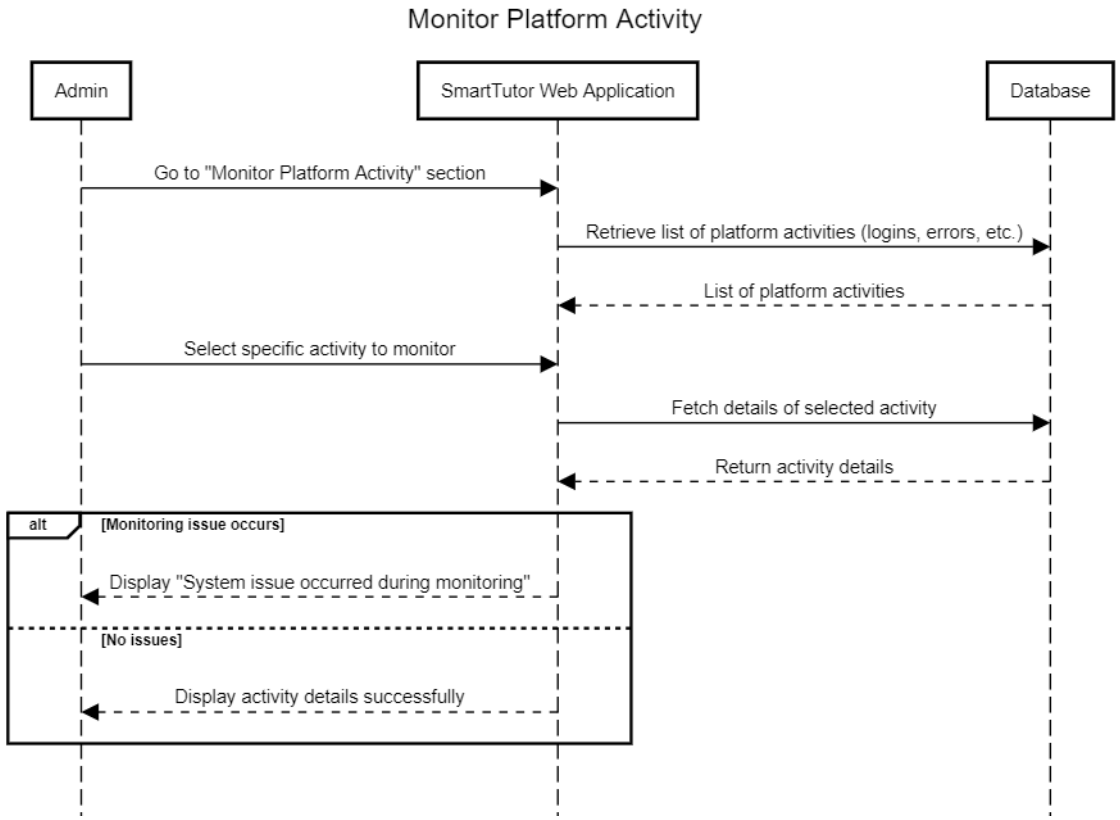

Figure 3.16: SSD (Monitor Platform Activity)

3.4.3.12 SSD 12: Manage Payment

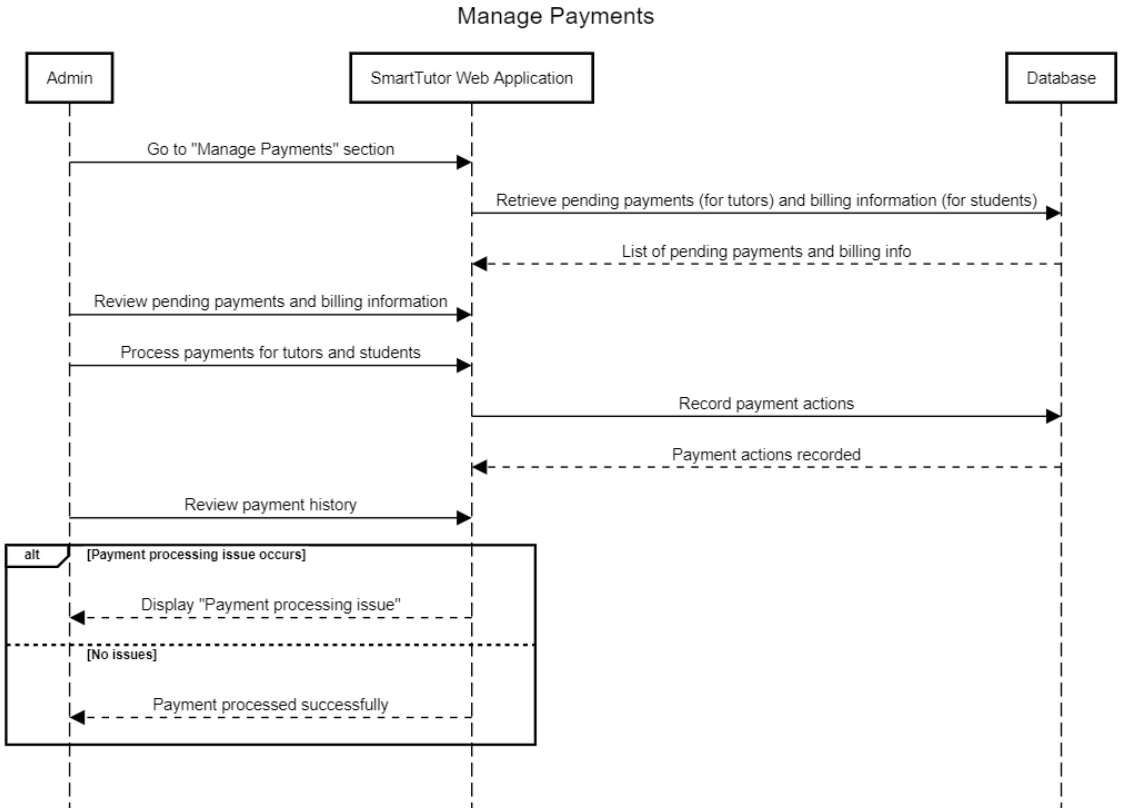

Figure 3.17: SSD (Manage Payment)

3.4.3.13 SSD 13: Send Notification

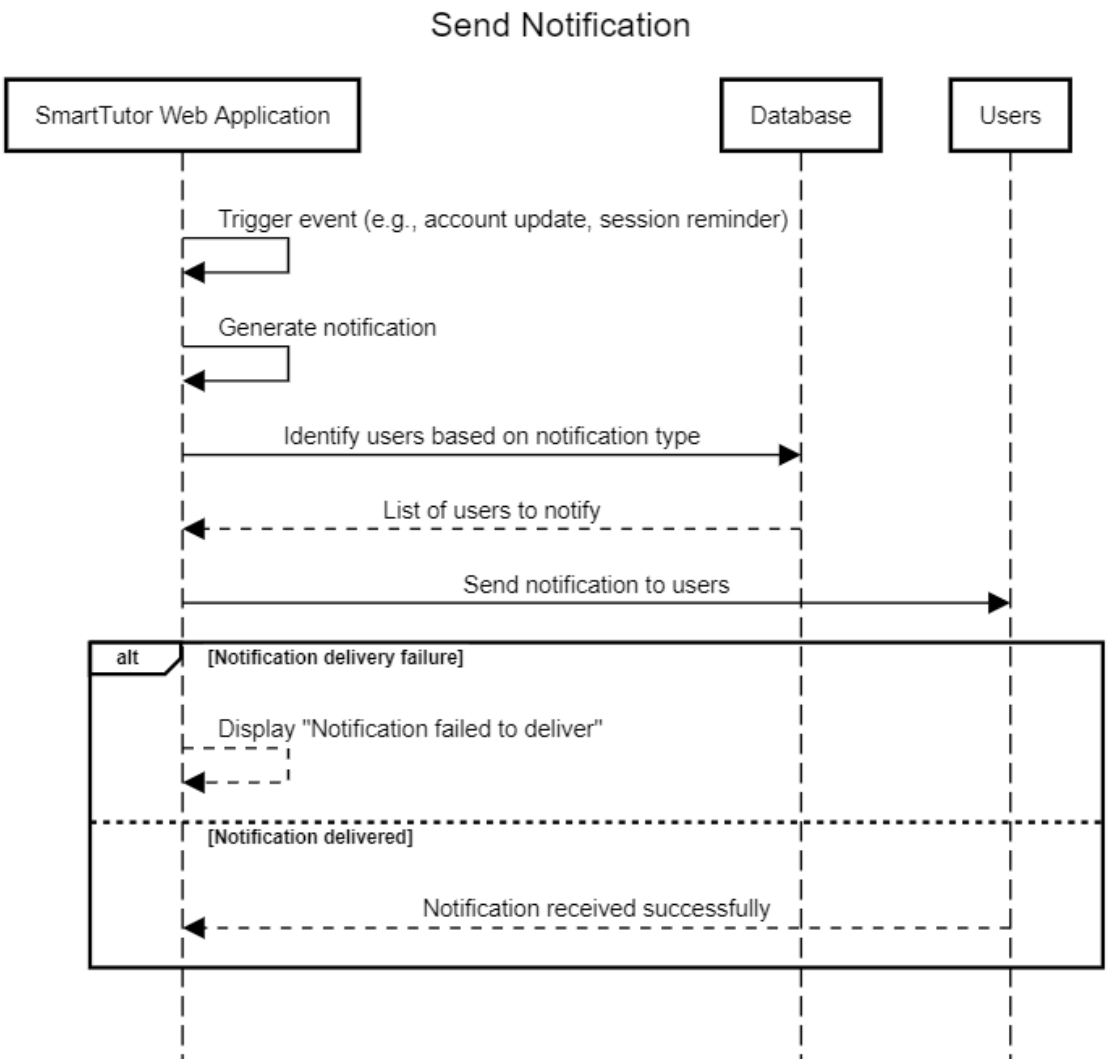

Figure 3.18: SSD (Send Notification)

3.4.3.14 SSD 14: User Login

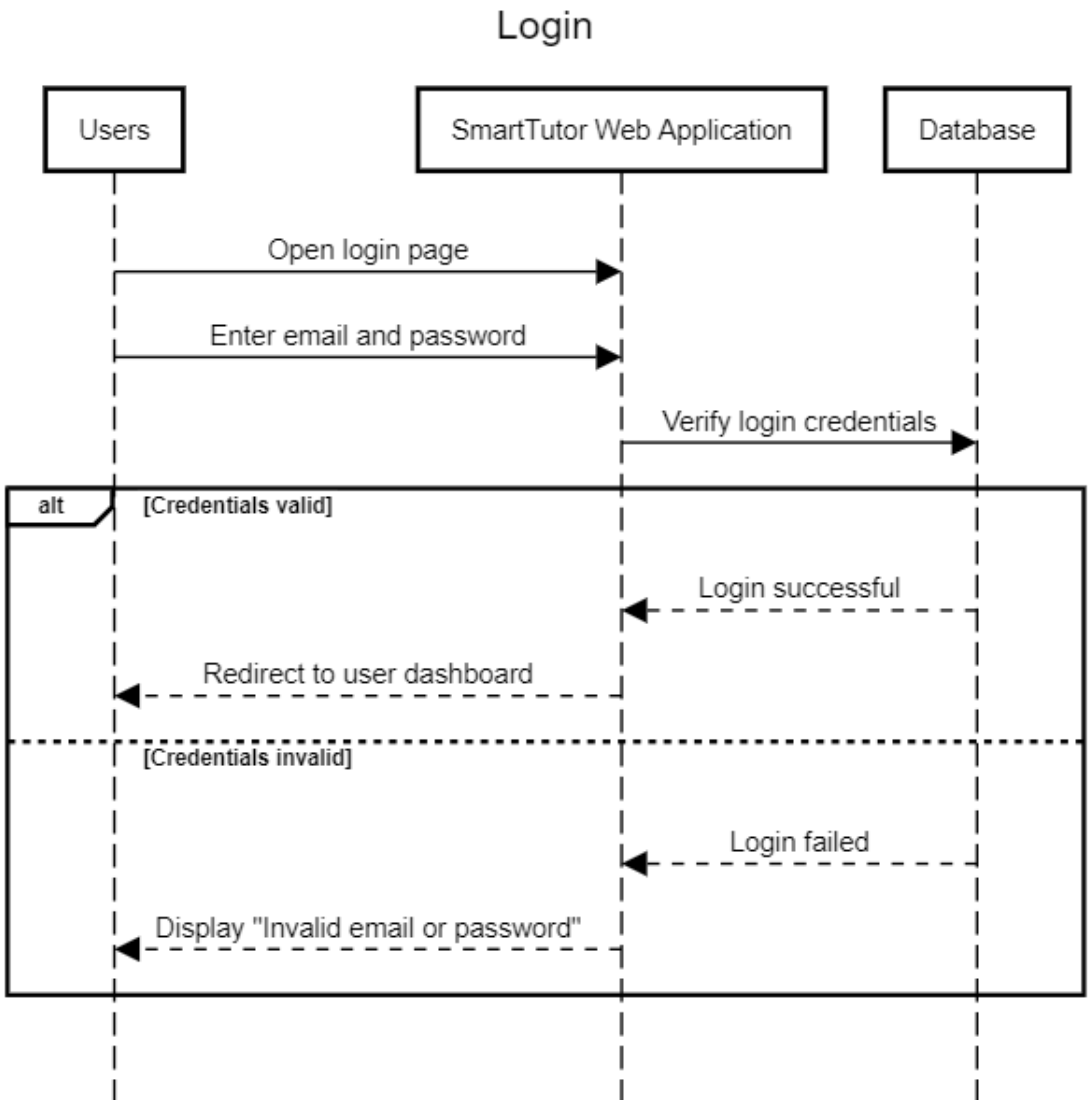

Figure 3.19: SSD (User Login)

3.4.4 State Transition Diagram

This state transition diagram represents the dynamic behavior of the SmartTutor system because, although it describes all those states to which different elements of the system may change, it also indicates transitions from one to another. It actually shows graphically how the system might respond to various events or conditions and how the states change due to user interactions and backend processes. This diagram can especially be helpful in tracking complex functionalities, such as user sessions, account management, and notification handling. In detailing states and transitions, the state transition diagram serves to assist in identifying problem areas and optimizing processes to ensure smooth flow within the application.

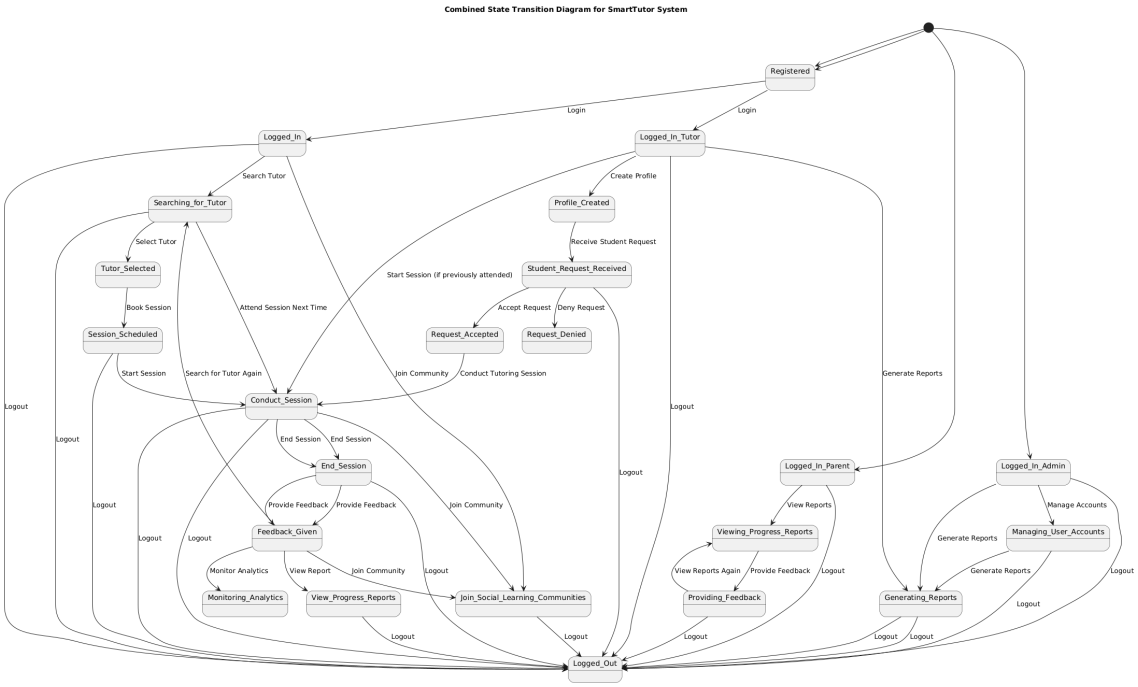

Figure 3.20: State Diagram

Chapter 4

Implementation and Testing [UPTO THE CURRENT ITERATION ONLY]

Give a general description of the functionality, context, and design of your project. Provide any background information if necessary.

4.1 Algorithm Design

Mention the algorithm(s) used in your project to get the work done with regards to major modules. Provide a pseudocode explanation regarding the functioning of the core features. Following are few examples of algorithms/pseudocode.

Example:

Algorithm 1 MHCF co-authorsBasedClustering

Input: n groups G_n where each group has set of papers (p_r)

Output: Set of system generated clusters/groups G_n

```
merge ← true
Flag ← false
While(merge=='true') do:
    merge ← false
        for i in range (0: len(G)-1):
            for j in range (i+1: len(G)):
                if (similarCoauthors(GiLco-authors, GjLco-authors) == true) then
                    Flag ← true
                Else (checkNameFragments(GiLco-authors, GjLco-authors) == true) then
                    Flag ← true
                if (Flag == true) then
                    Gi ← Gi U Gj
                    G ← G.pop(j)
                    merge ← true
                end if
            end for
        end for
end while
```

Figure 4.1: Example Of Algorithm Design

4.2 External APIs/SDKs

Describe the third-party APIs/SDKs used in the project implementation in the following table. Few examples of APIs are provided in the table.

API and version	Description	Purpose of usage	API endpoint/function/class used
Stripe (version 2020-08-27)	Credit Card payment integration	Sandbox used orders payment	stripe.paymentMethods.create
Cloudinary	Image and Video management	Uploading Product Images	https://api.cloudinary.com/v1

4.3 Testing Details

Once the system has been successfully developed, testing has to be performed to ensure that the system working as intended.

4.3.1 Unit Testing

Each unit test is designed to test a specific function or method independently from other components, helping to identify issues directly related to the functionality being tested.

Following is the example of Unit testing:

Test case ID	Test Objective	Precondition	Steps	Test data	Expected result	Post-condition	Actual Result	Pass/fail
TC001	Verify admin login with username and password	Admin should be registered with valid email and password before login.	Click on Login button Enter valid username and password	Email-id: abc@xyz.com Password: Xyz123	System displays Admin homepage	Admin should be kept logged in until logout.	As Expected,	Pass

Figure 4.2: Example for Unit Testing

Bibliography

[1] A Kolyshkin and S Nazarovs. Stability of slowly diverging flows in shallow water. *Mathematical Modeling and Analysis*, 2007.

Appendix A

Appendices

A.1 Appendix A

A.1.1 Use Case Diagram example (Online Shopping System)

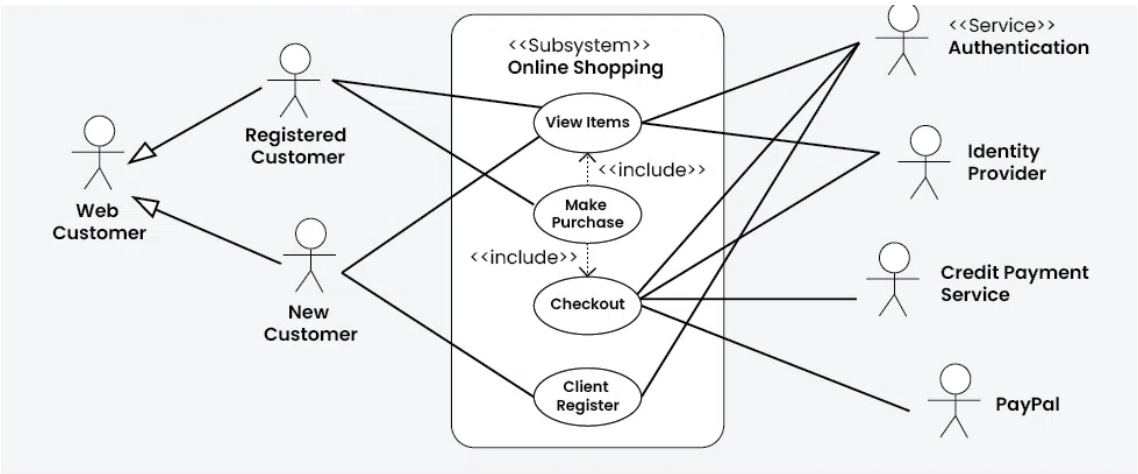

Figure A.1: Use Case Diagram for the Online Shopping System

A.1.2 Detail Use Case Example

ID	#1
Name	Overview elements
Short Description	All elements are shown in a list, where the user can set different filters.
Goal	Displaying elements in a changeable view.
Preconditions	The user got access to the system and is logged in. The user got the right to see schedules.
Success End Condition	The correct elements (filter and sorting) are displayed.
Fall End Condition	Elements not matching the criteria are displayed.
Stakeholder	Customer manager
Trigger	Login in as customer manager (because overview is on the starting screen for this role).
Normal Flow	1. The list of all elements matching default criteria are shown. 2. The user changes the filter criteria. 3. The user presses the Filter button. 4. The list shows all matching elements. Optional: 5. The user presses the Clear button. 6. The list of all elements matching default criteria are shown.
Alternative Flows	With click on the column headers, the list sorting can be changed. Different sorting for the different columns is described in the table below.
Includes	Login
Frequency of Use	About 50 times per day
Constraints and Special Requirements	None
Assumptions	None
Notes and Issues	None

Figure A.2: Detail Use Case Example

A.1.3 Event-Response Table for a Highway Intersection

Event	System State	Response
Road sensor detects vehicle entering left-turn lane.	Left-turn signal is red. Cross-traffic signal is green.	Start green-to-amber countdown timer for cross-traffic signal.
Green-to-amber countdown timer reaches zero.	Cross-traffic signal is green.	1. Turn cross-traffic signal amber. 2. Start amber-to-red countdown timer.
Amber-to-red countdown timer reaches zero.	Cross-traffic signal is amber.	1. Turn cross-traffic signal red. 2. Wait 1 second. 3. Turn left-turn signal green. 4. Start left-turn-signal countdown timer.
Pedestrian presses a specific walk-request button.	Pedestrian sign is solid Don't Walk. Walk-request countdown timer is not activated.	Start walk-request countdown timer.
Pedestrian presses walk-request button.	Pedestrian sign is solid Don't Walk. Walk-request countdown timer is activated.	Do nothing.
Walk-request countdown timer reaches zero plus the amber display time.	Pedestrian sign is solid Don't Walk.	Change all green traffic signals to amber.
Walk-request countdown timer reaches zero.	Pedestrian sign is solid Don't Walk.	1. Change all amber traffic signals to red. 2. Wait 1 second. 3. Set pedestrian sign to Walk. 4. Start don't-walk countdown timer.

Figure A.3: Example of Event Response Table

A.1.4 Story Board Example For Android App

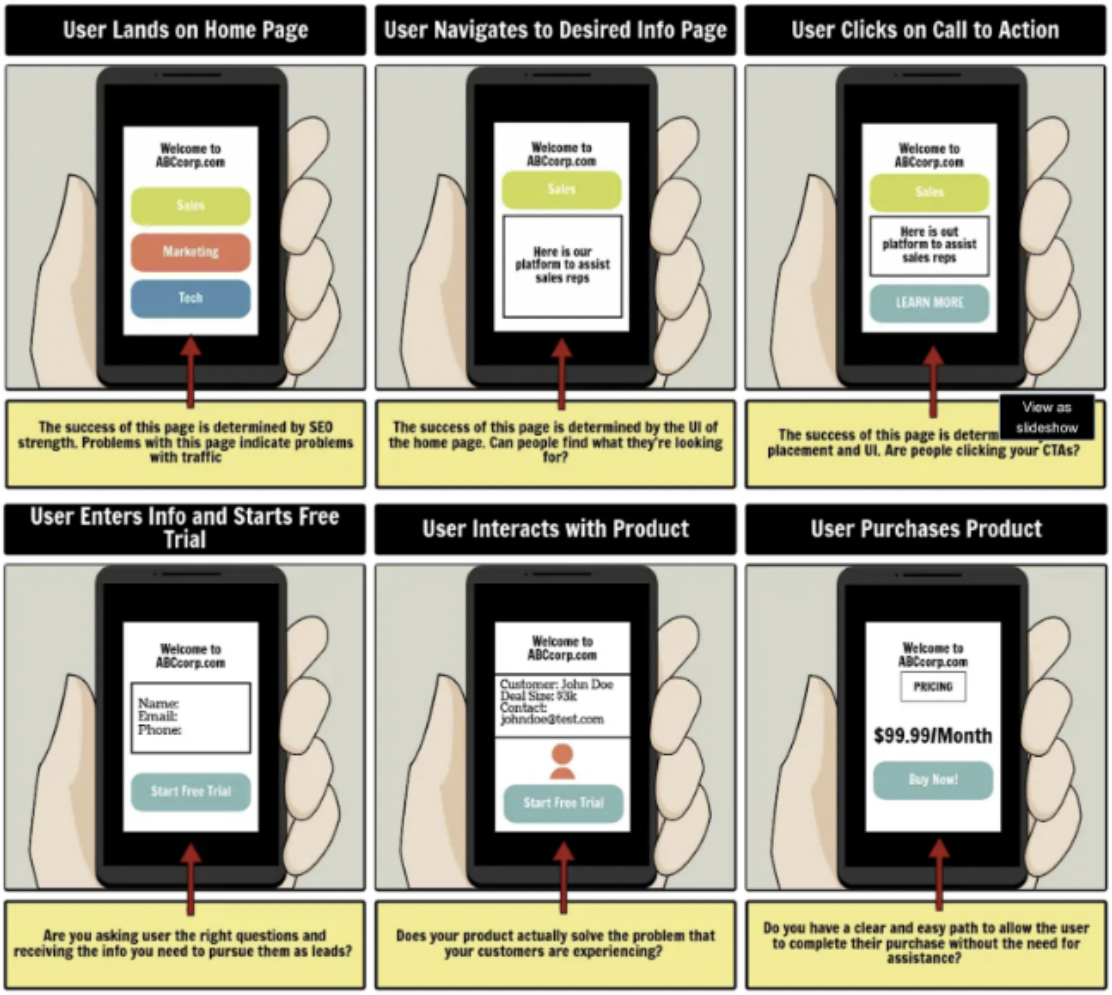

Figure A.4: Example of Story Board

A.2 Appendix B

A.2.1 Domain Model Example For Online Shopping

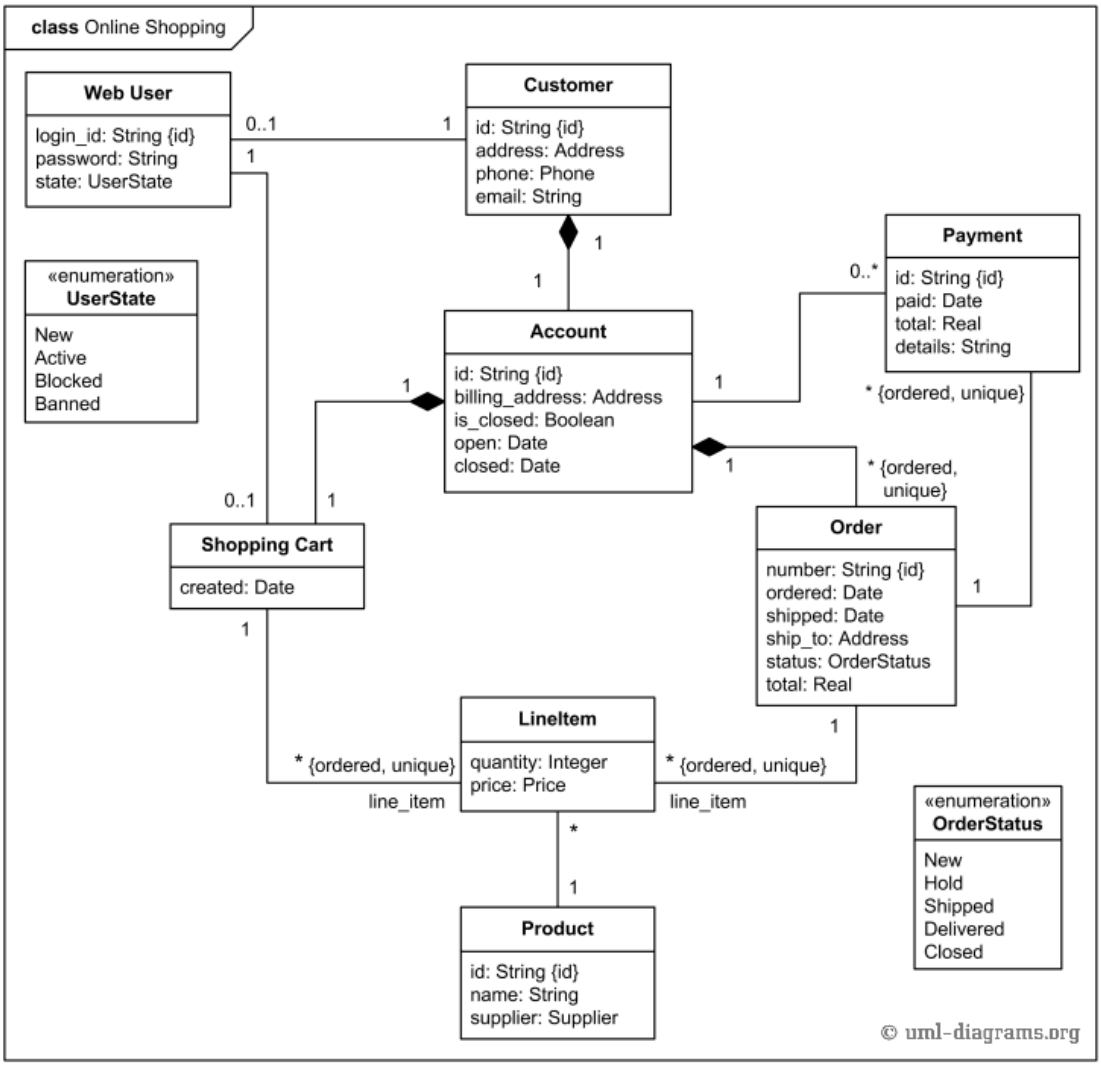

Figure A.5: Domain Model Example For Online Shopping Application

A.3 Appendix C

A.3.1 Box And Line Example For Online Shopping

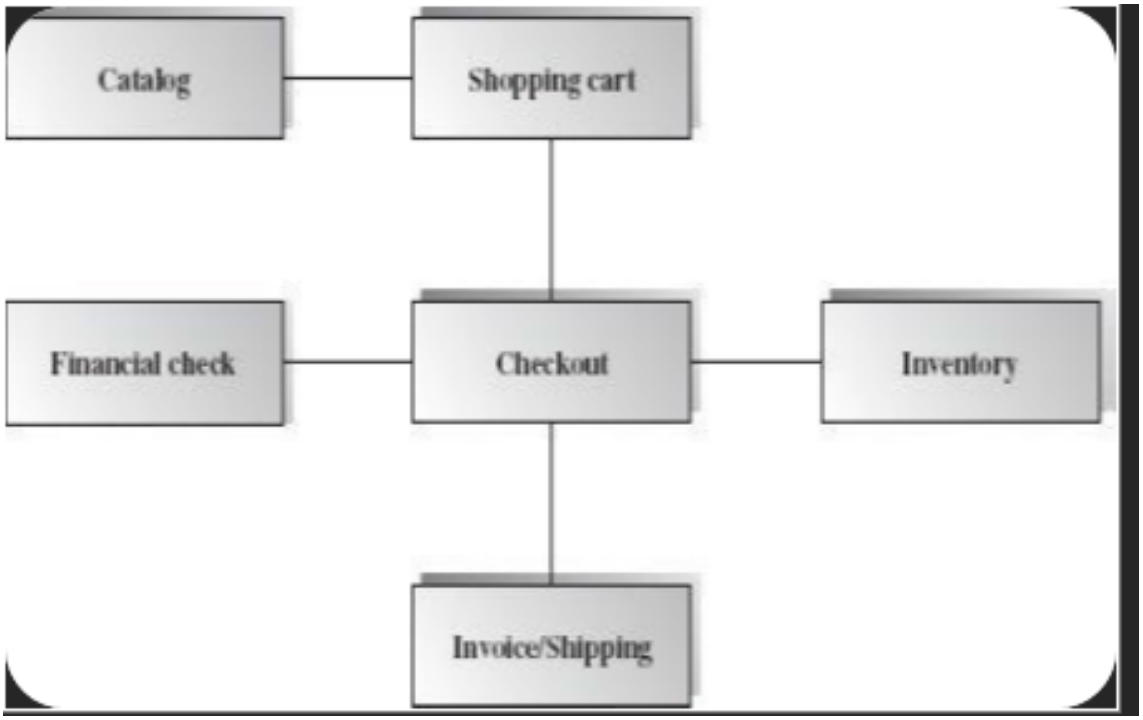

Figure A.6: Box and Line Diagram For Online Shopping Application

A.3.2 Architecture Pattern Example For Online Shopping

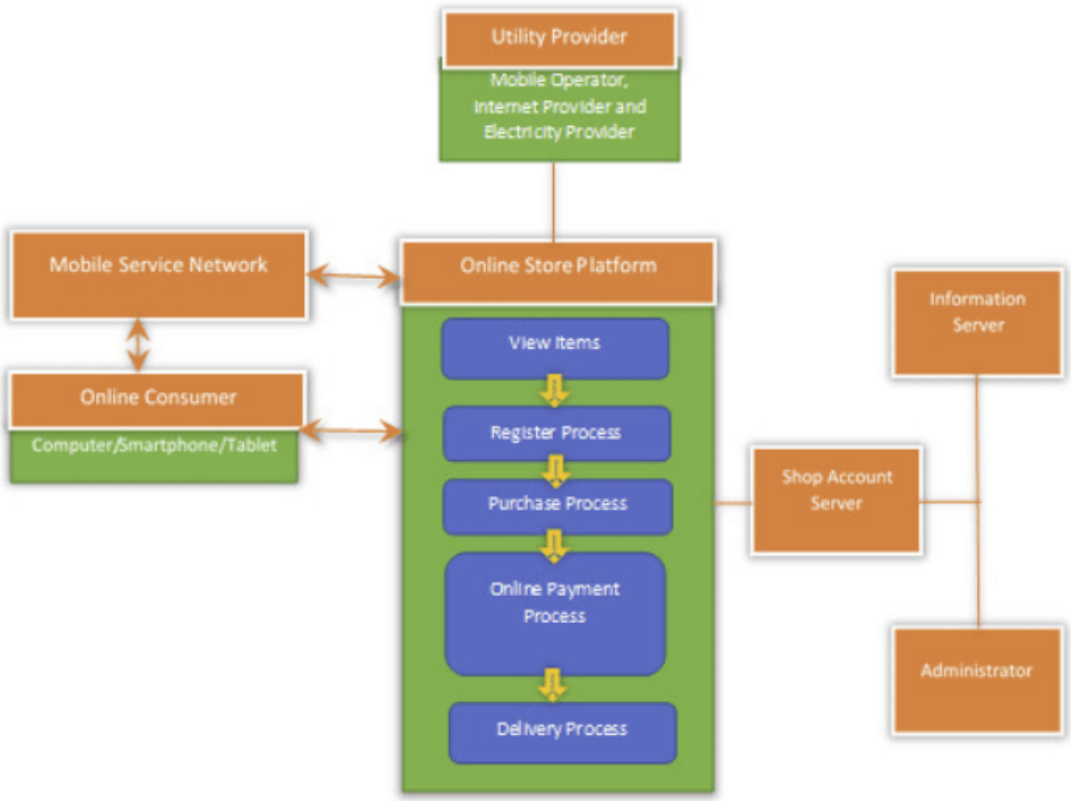

Figure A.7: Architecture Pattern For Online Shopping Application

A.4 Appendix D

A.4.1 Activity Diagram

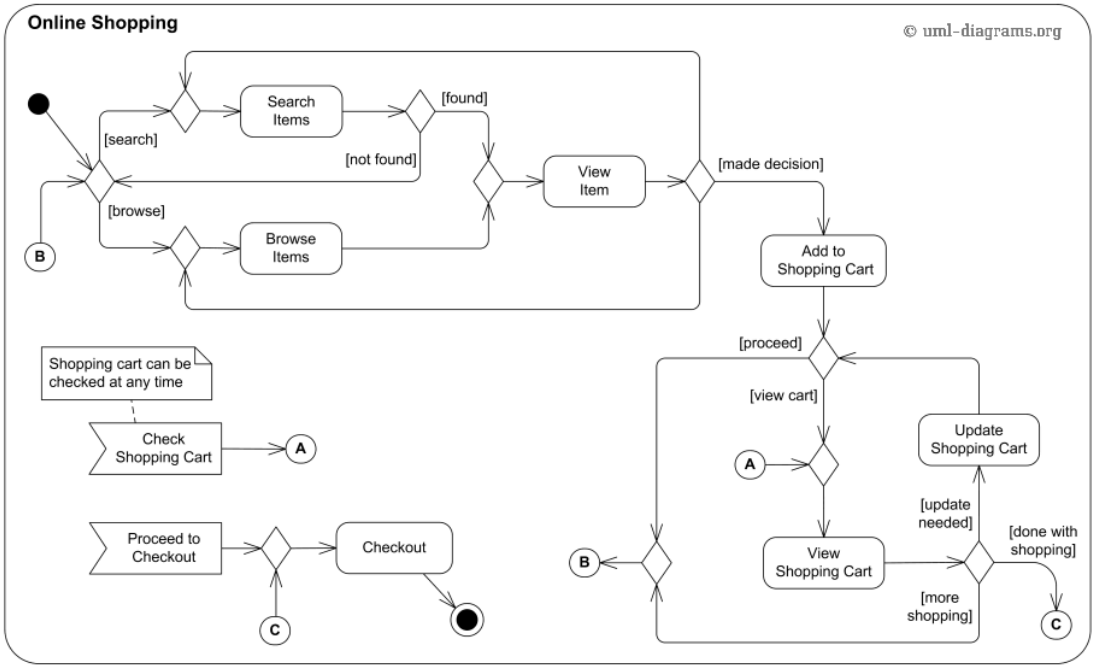

Figure A.8: Activity Diagram For Online Shopping Application

A.4.2 Class Diagram

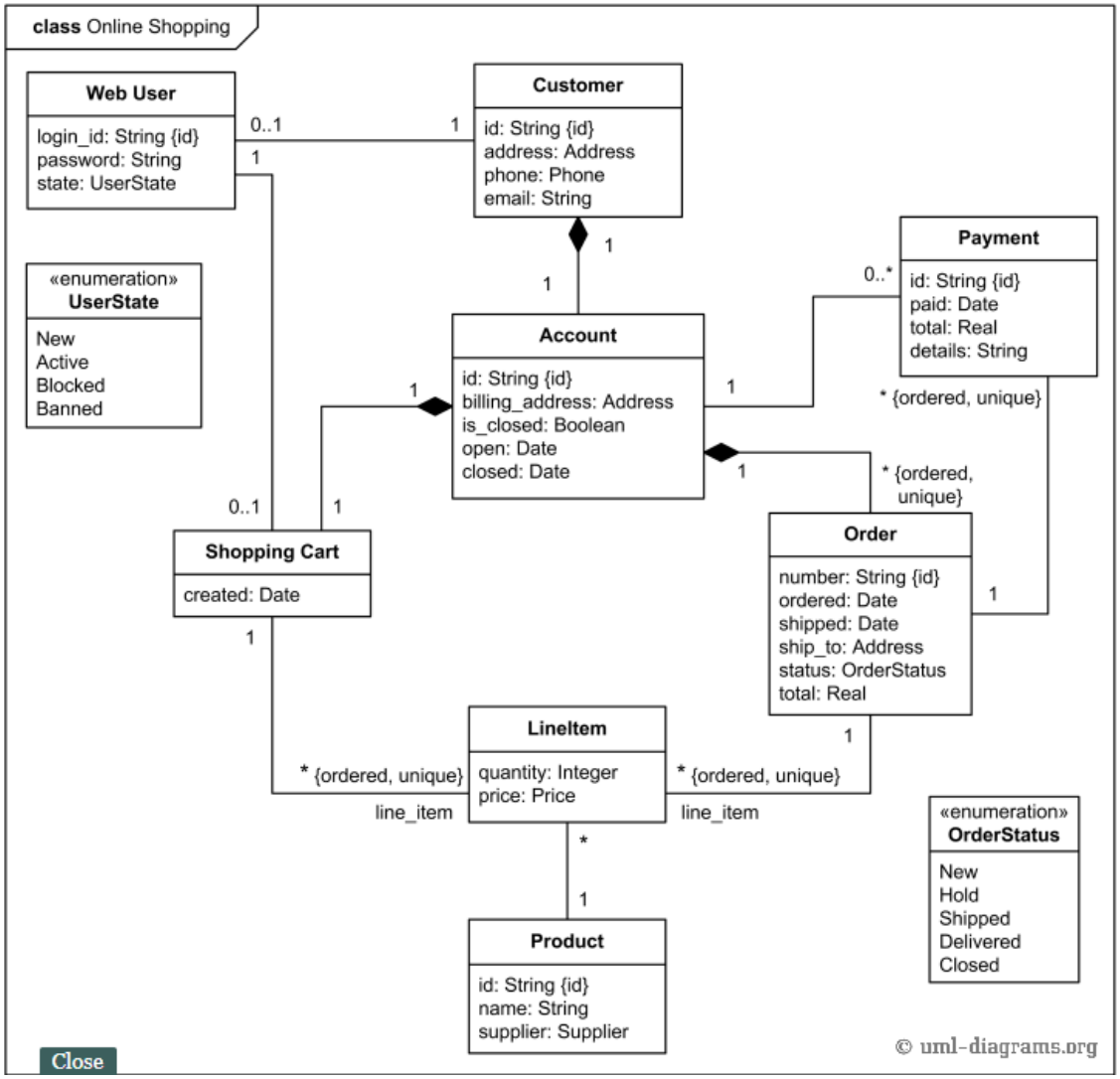

Figure A.9: Class Diagram For Online Shopping Application

A.4.3 Sequence Diagram

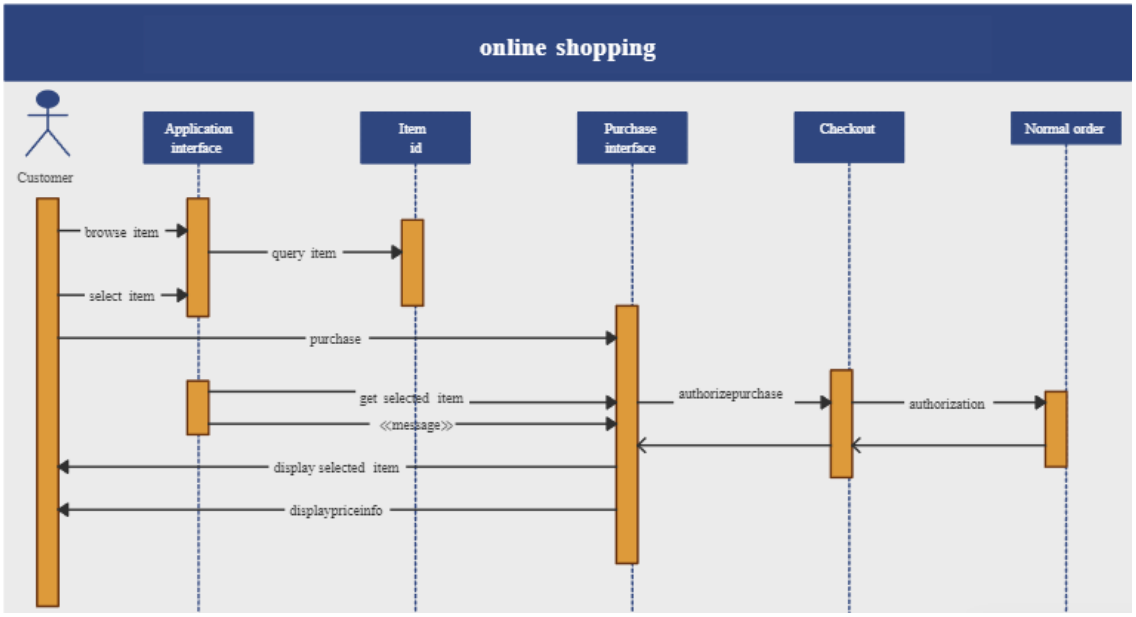

Figure A.10: Sequence Diagram For Online Shopping Application

A.4.4 State Transition Diagram

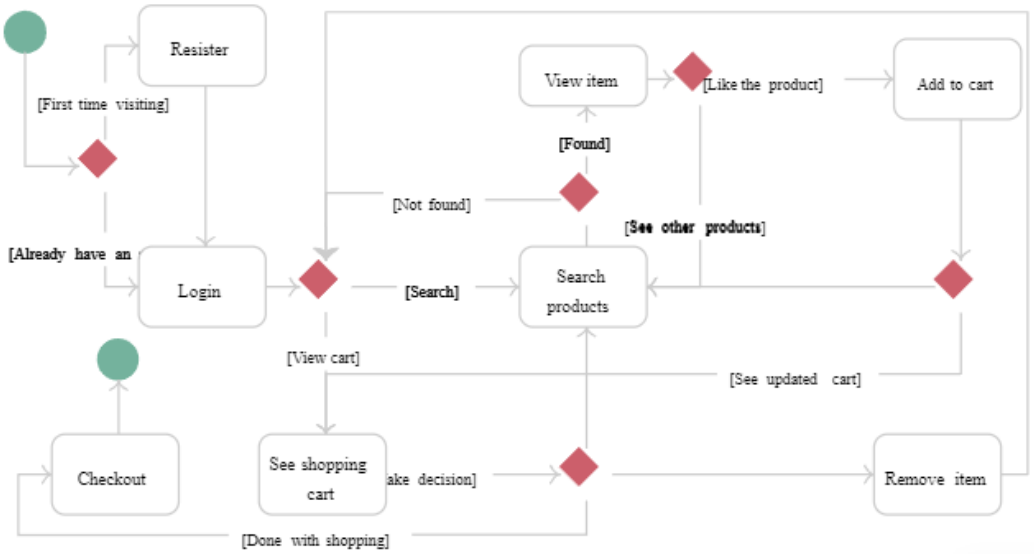

Figure A.11: State Transition Diagram For Online Shopping Application

A.4.5 Data Flow Diagram

Data Flow Diagram

Data flow diagram symbols, symbol names, and examples

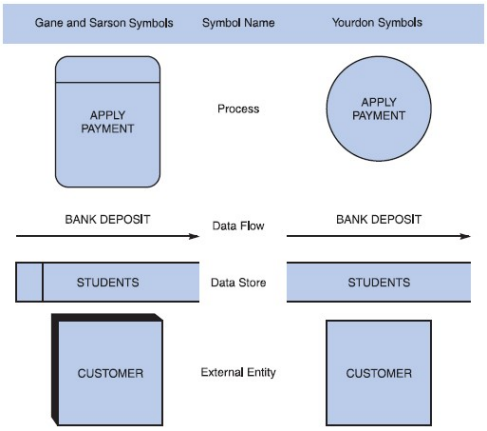

Figure B-5 Data flow diagram symbols, symbol names, and examples of the Gane and Sarson and Yourdon symbol sets.

Guidelines for Drawing DFDs

Step 1: Draw a Context Diagram: The first step in constructing a set of DFDs is to draw a context diagram. A **context diagram** is a top-level view of an information system that shows the system's boundaries and scope. Data stores are not shown in the context diagram because they are contained within the system and remain hidden until more detailed diagrams are created.

Example

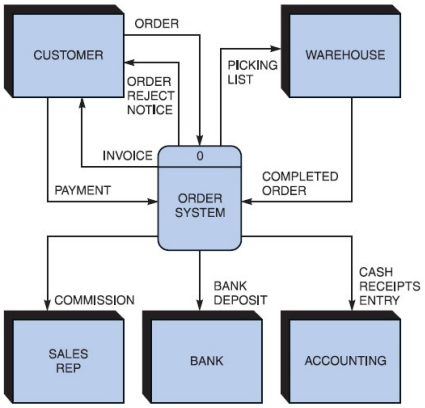

Figure B-6 Context diagram DFD for an order system.

Step 2: Draw a Diagram 0 DFD: To show the detail inside the black box, you create DFD diagram 0. **Diagram 0** zooms in on the system and shows major internal processes, data flows, and data stores. Diagram 0 also repeats the entities and data flows that appear in the context diagram. When you expand the context diagram into DFD diagram 0, you must retain all the connections that flow into and out of process 0.

Example

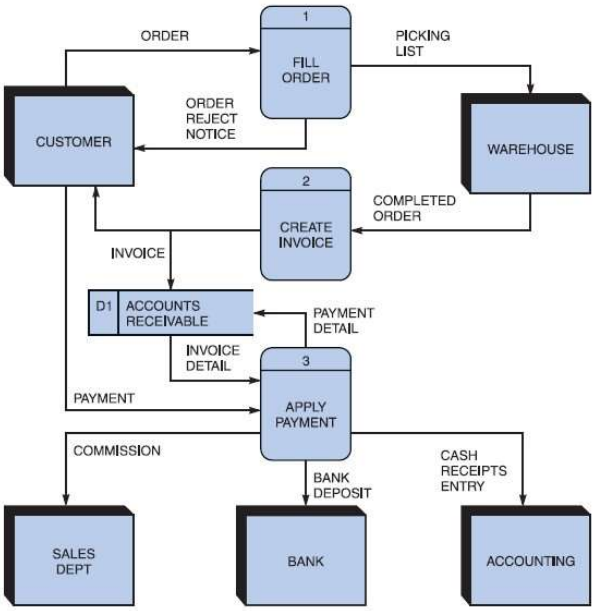

Figure B-7 Diagram 0 DFD for the order system.

Step 3: Draw the Lower-Level Diagrams:

To create lower-level diagrams, you must use leveling and balancing techniques. **Leveling** is the process of drawing a series of increasingly detailed diagrams, until all functional primitives are identified.

Leveling Example

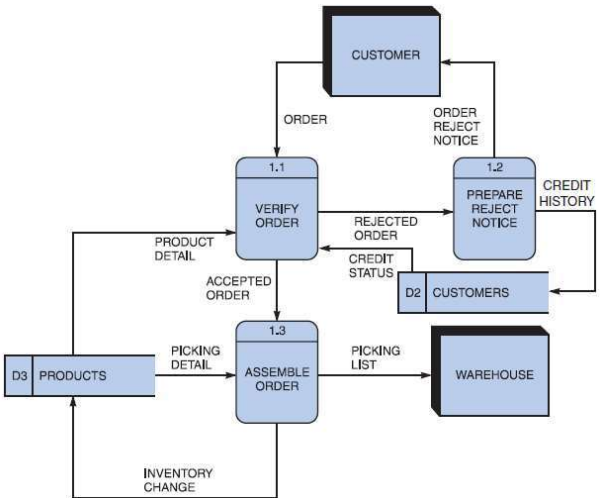

Figure B-8 Diagram 1 DFD shows details of the FILL ORDER process in the order system.