CEBU INSTITUTE OF TECHNOLOGY
UNIVERSITY

IT342-G1
SYSTEMS INTEGRATION AND ARCHITECTURE 1

FUNCTIONAL REQUIREMENTS SPECIFICATION (FRS)

Project Title: User Registration & Authentication SRS

Prepared By: James Michael Restauro Siton

Date of Submission: February 6, 2026

Version: 1.0

Table of Contents

1. Introduction

1.1. Purpose

The purpose of the System is to manage and control user access by verifying user identities. As its initial functionality, the System provides user authentication features that allow users to register, log in, view their profile, and log out. These features ensure secure access and proper handling of user sessions within the System.

This authentication functionality serves as the foundation for the System, which is expected to expand in the future to include additional features and services.

1.2. Scope

The System currently focuses on basic user authentication processes. It allows users to create an account, authenticate themselves through login, access their profile information after authentication, and securely log out of the System.

Functions such as password recovery, account management, role-based access, and other advanced features are outside the scope of the current implementation and may be included in future versions of the System.

1.3. Definitions, Acronyms, and Abbreviations

- **System** – The application being developed, which currently focuses on user authentication
- **User** – An individual who interacts with the System
- **Guest User** – A user who is not logged in
- **Authenticated User** – A user who has successfully logged in to the System
- **Authentication** – The process of verifying a user's identity
- **Session** – The period during which a user is logged in to the System

2. Overall Description

2.1. System Perspective

The System is designed as a modular application where user authentication is implemented as the initial core component. This authentication module can operate independently or be integrated into a larger system as development continues.

The System manages transitions between guest users and authenticated users while ensuring secure access to protected features.

2.2. User Classes and Characteristics

Guest User

● Has not logged in to the System
● Can register for an account
● Can log in using valid credentials
● Cannot access protected features such as profile viewing

Authenticated User

● **Has successfully logged in to the System**
● **Can view personal profile information**
● **Can log out to end the session**
● **Has access only to features allowed after authentication**

2.3. Operating Environment

The System operates in the following environment:
Hardware: Desktop computers, laptops, or mobile devices (Android)
Backend: Spring Boot (Java)
Frontend: ReactJS
Database: MySQL

2.4. Assumptions and Dependencies

● Users have access to a stable internet connection
● Users provide valid information during registration
● The System depends on a database for storing user data
● Basic security mechanisms, such as password encryption, are implemented

3. System Features and Functional Requirements

Describe each major feature of the system and its functional requirements.

3.1. Feature 1: User Registration

Description:
● The System allows a guest user to create a new account by providing the required registration information.

Functional Requirements:
● The System shall allow guest users to register an account
● The System shall validate user input before account creation
● The System shall store user information securely

3.2. Feature 2: User Log In

Description:
● The System allows registered users to authenticate themselves using valid credentials.

Functional Requirements:
● The System shall allow users to log in using registered credentials
● The System shall verify credentials before granting access
● The System shall change the user state from guest user to authenticated user upon successful login

3.3. Feature 3: View Profile

Description:
● The System allows authenticated users to view their profile information.

Functional Requirements:
● The System shall restrict profile access to authenticated users only
● The System shall display stored profile information
● The System shall prevent guest users from accessing profile data

3.4. Feature 4: User Logout

Description:
● The System allows authenticated users to end their session securely.

Functional Requirements:
● The System shall allow authenticated users to log out
● The System shall terminate the active user session upon logout
● The System shall return the user to guest user status after logout

4. Non-Functional Requirements

Security:
● The System shall protect user credentials from unauthorized access
● Only authenticated users may access protected features

Usability:
● The System shall be easy to navigate and understand
● Authentication processes shall be simple and clear for users

Performance:
● The System shall respond to registration and login requests within an acceptable time

Reliability:
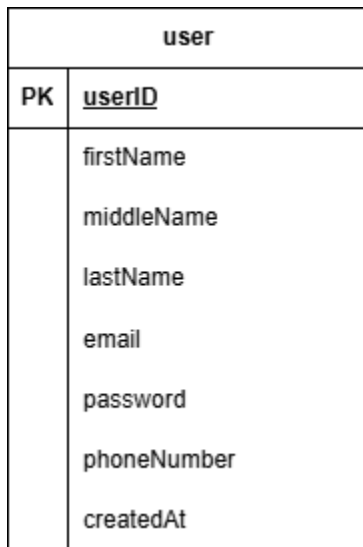● The System shall handle multiple users consistently
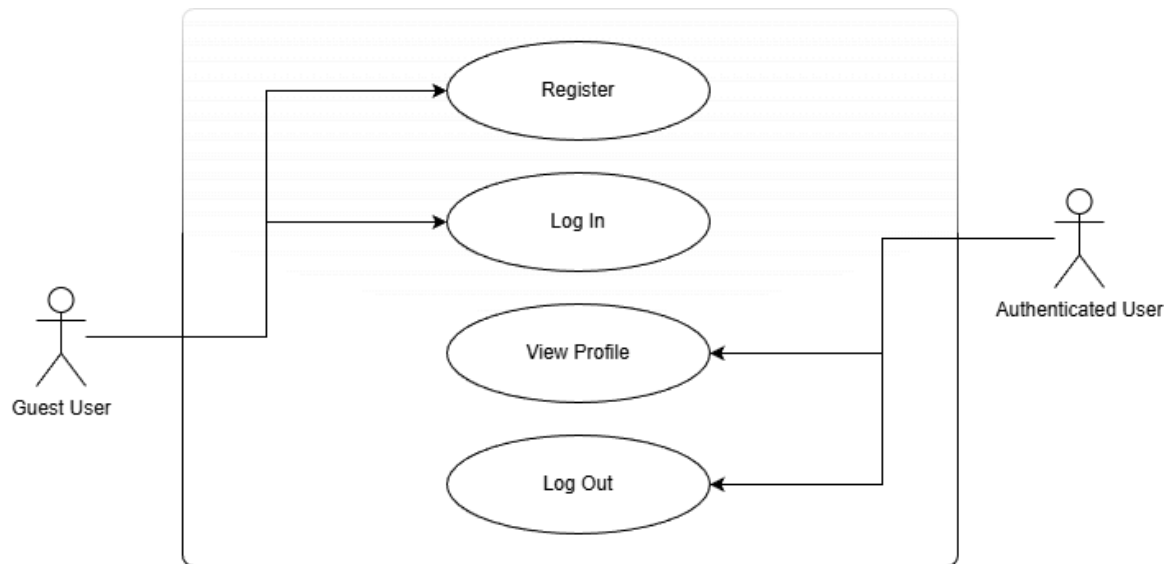● User sessions shall remain stable until logout

5. System Models (Diagrams)

Insert the necessary diagrams for the system:
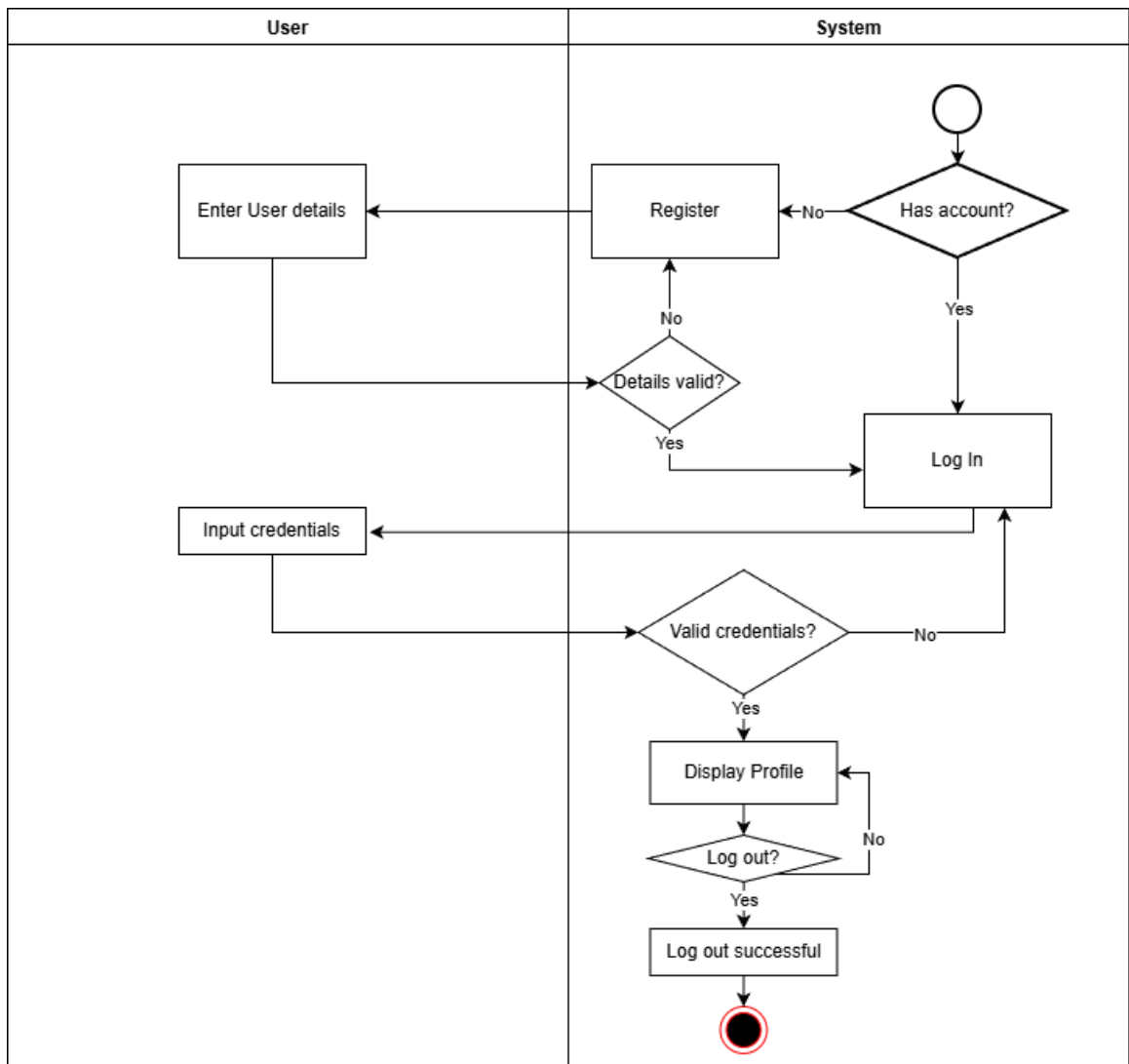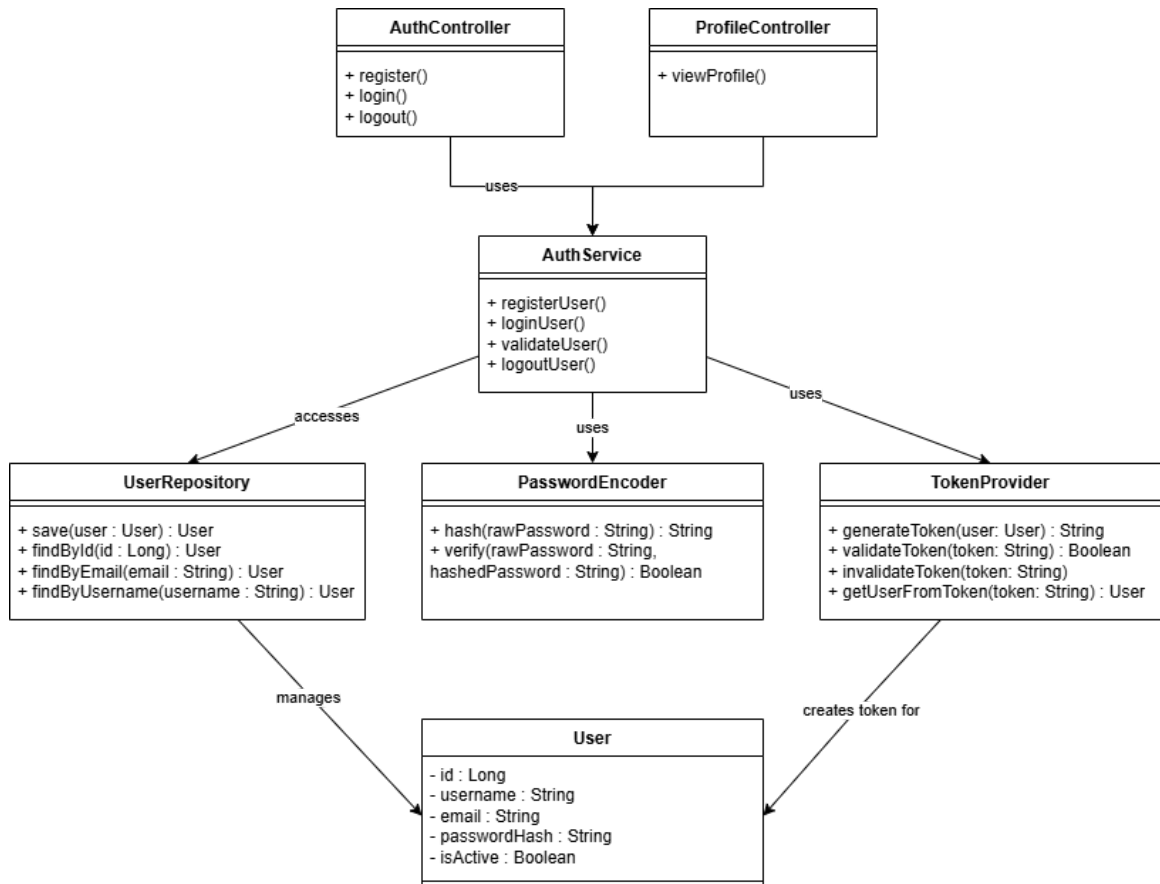
5.1. ERD

user	
PK	**userID**
	firstName
	middleName
	lastName
	email
	password
	phoneNumber
	createdAt

5.2. Use Case Diagram

5.3. Activity Diagram

5.4. Class Diagram

```
                    ┌──────────────────────┐        ┌──────────────────────┐
                    │    AuthController     │        │   ProfileController   │
                    ├──────────────────────┤        ├──────────────────────┤
                    │ + register()          │        │ + viewProfile()       │
                    │ + login()             │        │                       │
                    │ + logout()            │        │                       │
                    └──────────────────────┘        └──────────────────────┘
```

uses

```
                              ┌──────────────────────┐
                              │      AuthService      │
                              ├──────────────────────┤
                              │ + registerUser()      │
                              │ + loginUser()         │
                              │ + validateUser()      │
                              │ + logoutUser()        │
                              └──────────────────────┘
```

accesses   uses   uses

```
┌──────────────────────────────────┐  ┌──────────────────────────────────┐  ┌──────────────────────────────────────┐
│           UserRepository          │  │          PasswordEncoder          │  │             TokenProvider             │
├──────────────────────────────────┤  ├──────────────────────────────────┤  ├──────────────────────────────────────┤
│ + save(user : User) : User        │  │ + hash(rawPassword : String) : String │ │ + generateToken(user: User) : String  │
│ + findById(id : Long) : User       │  │ + verify(rawPassword : String,    │  │ + validateToken(token: String) : Boolean │
│ + findByEmail(email : String) : User │ │ hashedPassword : String) : Boolean │  │ + invalidateToken(token: String)       │
│ + findByUsername(username : String) : User │                            │  │ + getUserFromToken(token: String) : User │
└──────────────────────────────────┘  └──────────────────────────────────┘  └──────────────────────────────────────┘
```

manages   creates token for

```
                              ┌──────────────────────┐
                              │         User          │
                              ├──────────────────────┤
                              │ - id : Long           │
                              │ - username : String   │
                              │ - email : String      │
                              │ - passwordHash : String │
                              │ - isActive : Boolean  │
                              └──────────────────────┘
```
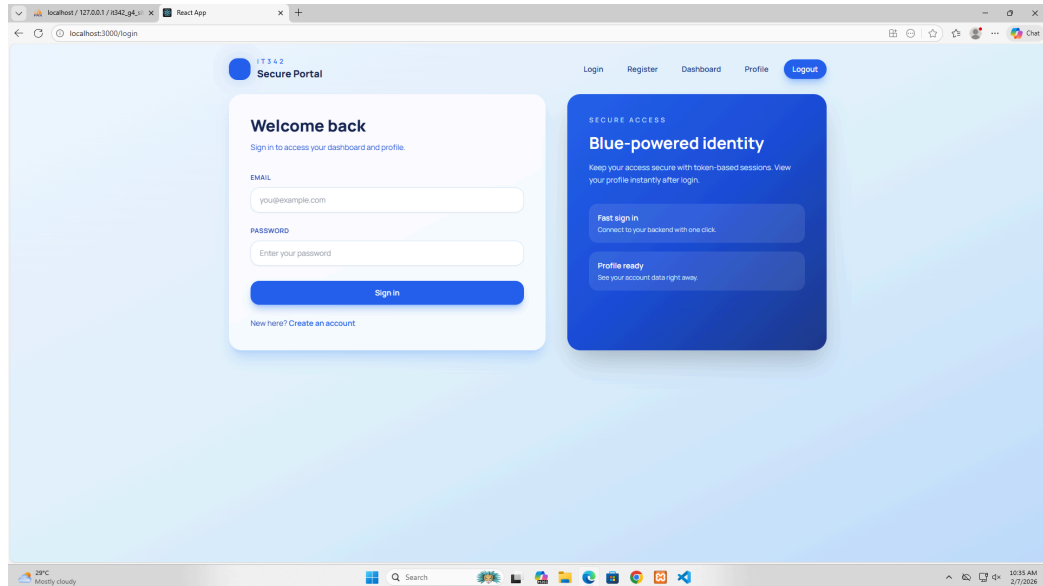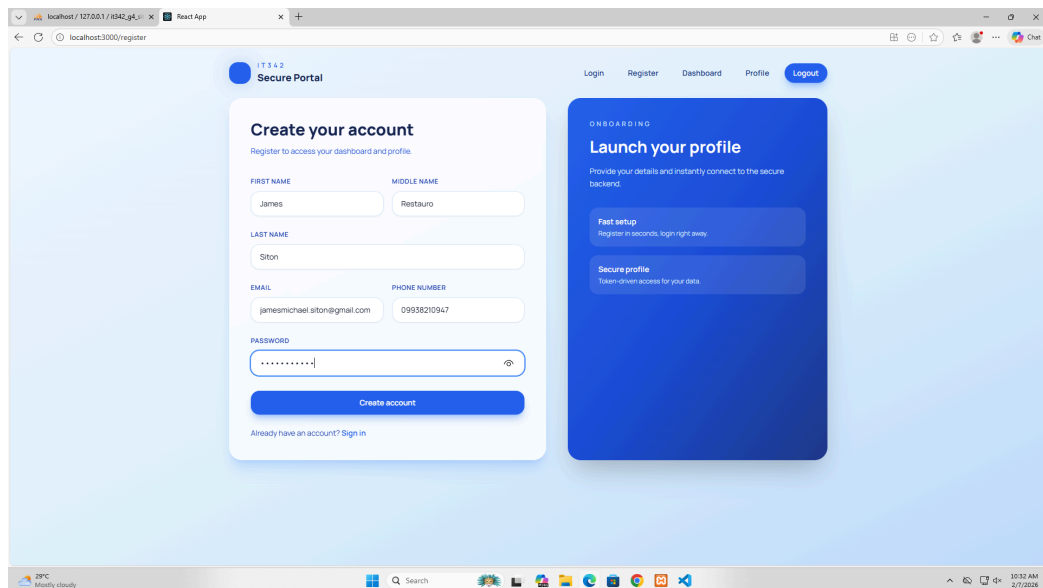
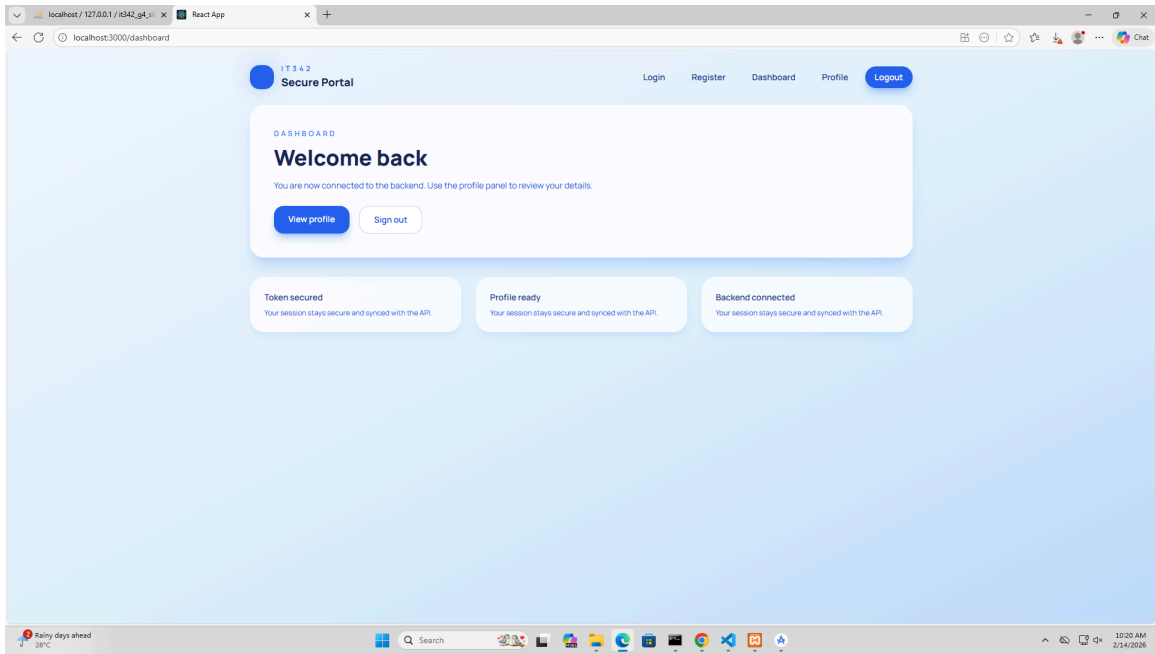## 5.5. Sequence Diagram
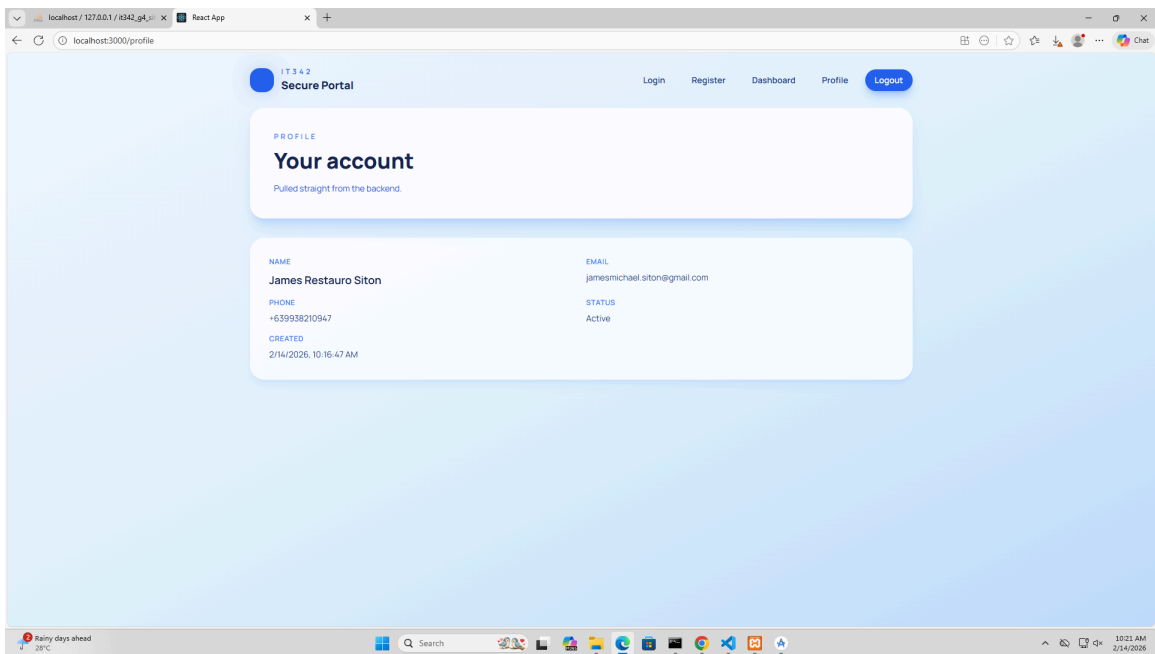
5.6. Screenshots from WebUI

Log In

Sign Up

Dashboard

View Profile

5.7. Screenshots from Mobile

Log In

Register

Dashboard

View Profile

6. Appendices

Include any additional information, references, or support materials.