Software Requirements Specification

for

Residential Security Aid ShiftStreamline Application

Version 1

Prepared by Team RSAShiftManage

Syracuse University CSE687

April 01, 2024

Table of Contents

Table of Figures

1 Introduction

1.1 Purpose

The Residential Security Aid ShiftStreamline Application's functional and non-functional requirements are described in this Software Requirements Specification (SRS) document. The purpose of this document is to provide guidance to the development team as they create an application that will increase the efficiency and efficacy of security management in residential communities. Its objectives are to provide a clear overview of capabilities for end users, a thorough blueprint for developers, and a point of reference for stakeholders.

1.2 Scope

The goal of the Residential Security Aid ShiftStreamline Application is to improve the operational dynamics of security in residential settings with a cutting-edge tool. Its emphasis on intuitive design, real-time situational awareness, and smooth communication between security personnel and management set it apart from traditional shift management systems. Because of its Object-Oriented Design base, the programme has a resilient, flexible, and scalable design that guarantees its long-term relevance and efficacy in tackling changing security concerns.

1.3 Definitions

Table 1 Acronyms and Definitions

BDD: Block Definition Diagram.
SRS: Software Requirements Specification.
UML: Unified Modeling Language.
V&V: Verification and Validation.
OOD: Object-Oriented Design
UI: User Interface
API: Application Programming Interface
RDBMS: Relational Database Management System
Database: A collection of data organized in a structured way.
Backend - A software application's backend is the portion in charge of handling data management and frontend request processing.

Frontend: A software application's portion that communicates with users directly and shows data.
User Experience (UX)- refers to the complete interaction between a user and a software programme, encompassing aspects such as visual design, usability, and accessibility.
Shift- is the time slot that an employee is assigned to work.
Schedule: A list of the days and hours when workers are expected to report for duty.
Availability: The hours and days that a worker is available for work.
Encapsulation: Sensitive data securely encapsulated, accessible only through defined interfaces. -
Abstraction: Simplifies scheduling complexity via engaging UI.
Inheritance: Promotes code reuse and system extensibility through reusable components.
Polymorphism: Enables flexible shift management by handling various scenarios dynamically

This section will provide clear definitions for technical terms, acronyms, and abbreviations used throughout the SRS to ensure consistency and avoid ambiguity.

1.4 References

1. IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications - Revision of IEEE Std 830-1993

2. Gamma, E., Helm, R., Johnson, R., & Vlissides, J.(1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.

3. Fowler, M.(2003). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition). Addison-Wesley Professional.

4. OWASP Top 10 - 2021. The Open Web Application Security Project's list of the ten most critical web application security risks, offering guidance on securing web applications against common threats.

1.5 Overview

The SRS will provide an overview of the application, including its functionality, intended use, and system restrictions, after the introduction. The ensuing sections will address certain needs, such as system attributes, user engagements, performance standards, and security procedures, guaranteeing a thorough foundation for the creation of the ShiftStreamline Application.

The ShiftStreamline Application establishes itself as a critical tool in the residential security landscape by combining real-time data with sophisticated scheduling algorithms and a user-centric design to fortify communities with unmatched operational intelligence and responsiveness. This introduction lays the groundwork for a thorough examination of the application's design, which aims to realise the idea of safer, better-managed living spaces.

2 Overall Description

2.1 Product Perspective

Figure 1 System Block Diagram shows the system overview, using a Unified Modeling Language (UML) Block Definition Diagram (BDD).

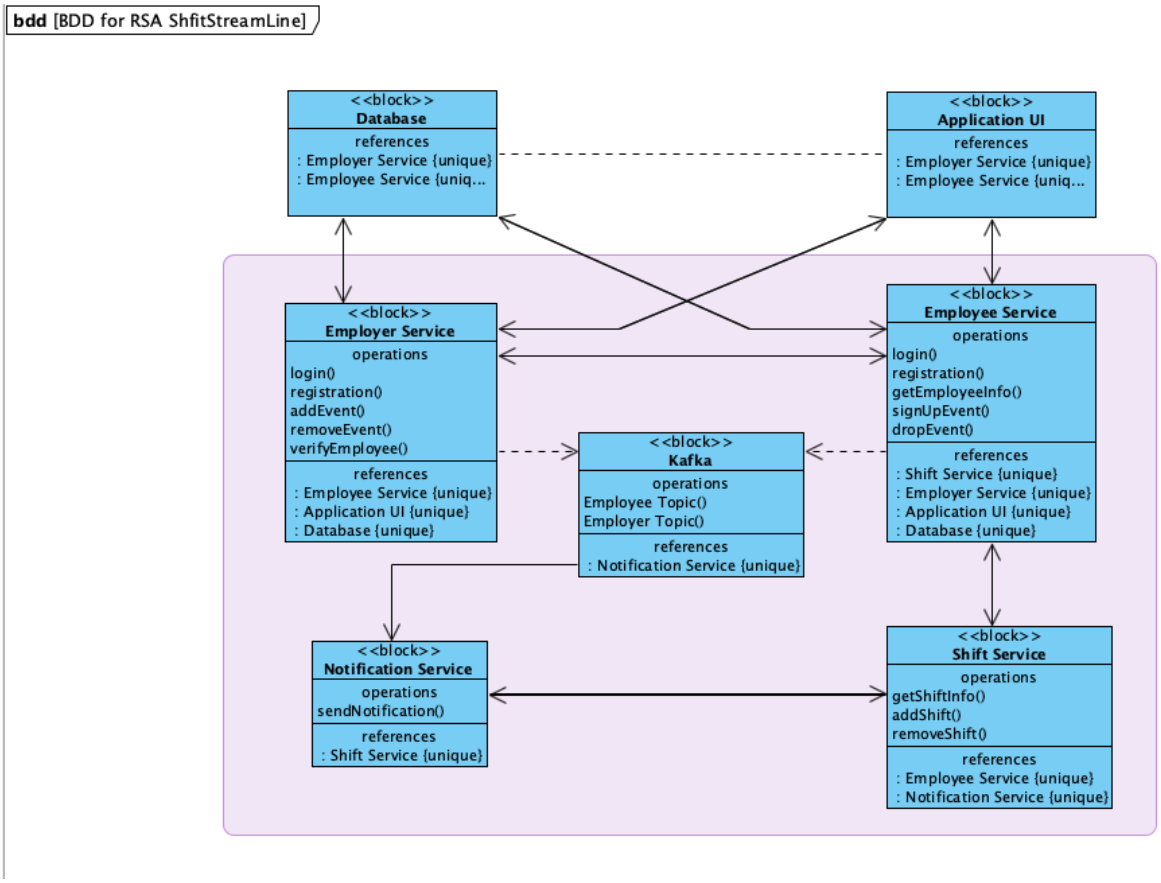

Figure 1 System Block Diagram

System Block Diagram for RSA ShiftStreamLine

The diagram shows the system architecture of RSA ShiftStreamLine, which consists of several interconnected blocks representing different services and components. The relationships between blocks show information flow and dependencies.

Blocks and their functions:

1. Employee Service:

Functions: Contains functions such as "login()", "registration()", "addEvent()", "removeEvent()" and "verifyEmployee()" .

References: uniquely links to API and database blocks, indicating that it communicates with those components.

2. Notification Service:

Functions: Consists of a single function "sendNotification()".

References: Has a unique reference to the Shift Service block.

3 . ShiftService:

Functions: Manages functions related to shifts, including "getShiftInfo()", "addShift()" and "removeShift()".

References: be individually connected to employer service and notification service blocks.

4. Kafka:

Features: No features listed.

References: Contains themes such as Worker Theme and Change Theme that can be used for queue and stream handling.

References: Interacts uniquely with the employer service, meaning it is the interface to the employer functions.

Relationships:

- The employer service acts as a central hub, which has two-way communication with the user interface of the application, indicating a continuous exchange of data.
- The notification service is positioned to act on data from the exchange service, probably notifying stakeholders of changes in the exchange .
- Kafka appears to act as a middleware, probably processing real-time data streams, showing a connection to the employer service.

2.2 Product Functions

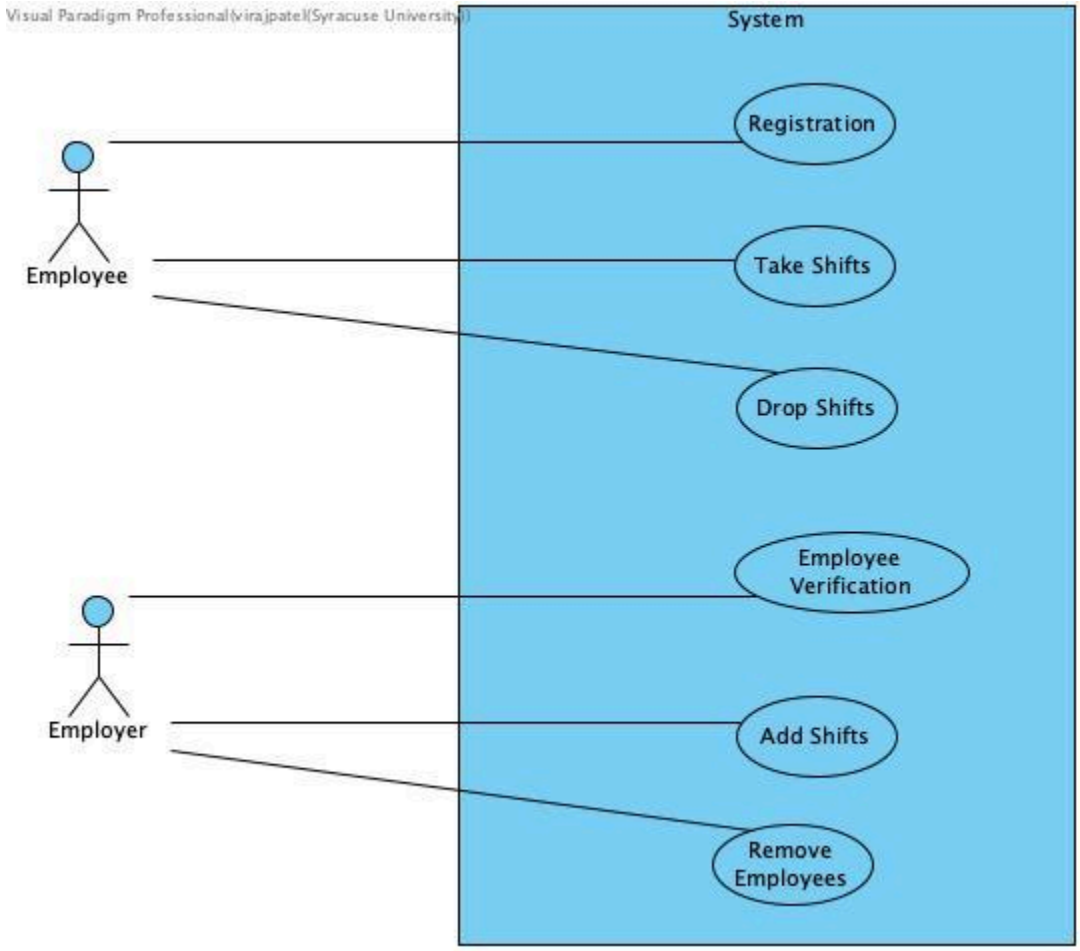

Figure 2 Use Case Diagram

Introduction

A use case diagram that describes how users and the system interact is provided in this section of the software requirements document. It shows the functional requirements of the system as well as the various ways that the 'Employee' and 'Employer,' two distinct users, will use it to achieve their objectives.

Actors

Employee: The user having the ability to sign up, accept and decline shifts, and go through the system's verification process.

Employer: This user is in charge of adding shifts, deleting employees from the system, and confirming employees.

Use Cases

Registration: In order to be identified as users, staff members must first register a new account in the system.

Take Shifts: Workers are permitted to accept any open shifts that the company may provide.

Drop Shifts: If necessary, employees may also choose to drop their allotted shifts.

Employee Verification: It is the duty of employers to confirm an employee's qualifications or job status in the system.

Add Shifts: Supervisors are able to designate and oversee shifts, allocating them to workers as required.

Remove Employees: Employers have the ability to take employees out of the system for administrative purposes, such terminating or changing the status of employment.

Functional Requirements

1.The system must enable staff members to create accounts.

2.The system must offer a way for workers to accept open shifts.

3.The system will allow workers to leave their shifts early.

4.The system must enable the employer to verify an employee's credentials.

5. The employer may add shifts to the schedule using the system.

6.The employer may remove workers from the system by using the system.

Non Functional Requirements

Usability: Employers and employees should be able to effortlessly navigate through the many capabilities of the system thanks to its user-friendly design.

Reliability: Employee registrations, shift assignments, and verification procedures must be handled by the system in a dependable manner.

Performance: There should be as little downtime or delays as possible in the system's handling of all use case actions.

Security: Carefully handled and kept sensitive employee data gathered throughout the registration and verification procedure is required.

2.3 Product Behaviors

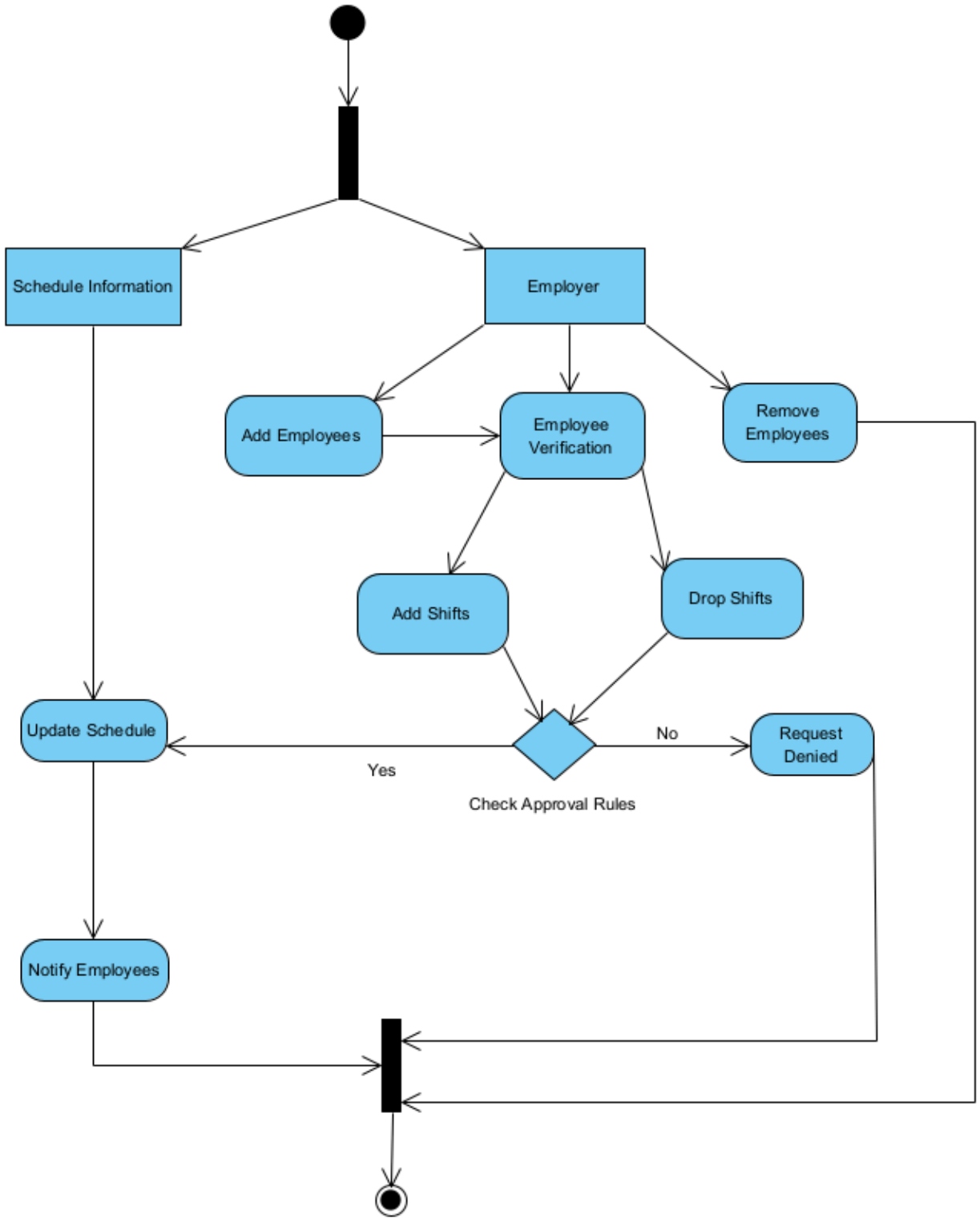

Figure 3 Enter Available Ingredients Activity Diagram

Introduction

The activity diagram, which shows the system's schedule management process flow, is described in detail in this section of the Software Requirements Specification document. The figure illustrates the sequential actions that a "Employer" takes to oversee shifts for workers, including shift verification, addition, removal, and updating, as well as the employees' subsequent communication of these changes.

Activity Diagram Overview

Steps in the process:

Start: The process begins when the schedule information is available or needs to be updated.

Employer: The employer is the main participant in the process.

Adding employees: The employer can add new employees to the system if you wish.

Employee verification: Before adding work shifts or personnel management, the data or authorization of the employee is verified.

Remove employees: The employer has the right to remove employees from the system.

Add shifts: The employer can add new shifts to the schedule.

Work Absences: Employees can request to suspend shifts, which leads to the next decision point.

Check approval rules. Conditionally checks if the exchange request can be accepted.

If so, the process continues. to update the schedule.

If not, the request is rejected and the schedule remains unchanged.

Update schedule: If the changes are approved, the system updates the schedule accordingly.

Notify employees: employees are notified of the updated schedule or other related . changes.

End: The process ends when messages are sent..

2.4 Product Requirements

Functional Requirements

1. Employee Registration:

By entering the required data, employees should be able to register themselves through the system.

A secure authentication method for employees to log in should be included in the system.

2. Employee Verification:

A mechanism for confirming employees' credentials and authorization levels should be built into the system.

Only authorised personnel should be able to access the system and carry out specific tasks.

3. Schedule Management:

Employers and administrators, for example, should be able to generate and modify employee schedules using the system.

The ability to add, remove, and change employee shifts within the schedule should be provided by the system.

4. Employee Management:

Users with permission should be able to add new staff members to the system.

The system ought to have the ability to deactivate or remove current employees from it.

5. Shift Management:

Permitted users should be able to add or remove assigned shifts from employees' schedules through the system.

Employees should be able to check the shifts they have been assigned and maybe request shift swaps or adjustments through the system.

6. Approval Rules and Validation:

To validate and approve schedule modifications and shift assignments, the system should include a set of established rules or policies.

The system should guard against unauthorised or incompatible changes and make sure schedule updates follow these guidelines.

7. Notification System:

A feature of the system should be the ability to alert staff members to any modifications or adjustments made to their shift assignments or schedules.

Email, in-app notifications, and other suitable channels are some possible ways to send notifications.

8. Request Handling:

The system ought to offer a mechanism for staff members to file requests pertaining to their work schedules or shifts (such as leave requests or requests for shift changes).

According to the established rules and norms, the system ought to include a procedure for reviewing, approving, or rejecting such requests.

9. Analytics and Reporting:

The system ought to produce reports and offer analytical insights concerning shift assignments, staff schedules, and other pertinent information.

The processes of workforce planning, resource allocation, and decision-making may benefit from the use of these reports and analytics.

10. User Roles and Permissions Management:

To handle user roles and permissions, the system needs a strong access control mechanism.

Administrators, supervisors, and employees, for example, should each have the proper access levels and rights within the system.

Non-functional Requirements

Performance:

The system should respond to requests for schedule updates, shift assignments, and personnel verification in a reasonable amount of time.

Efficient data retrieval and processing can be ensured by implementing suitable caching methods and optimisations.

Security:

To safeguard sensitive data and guarantee that only authorised users may access and take activities, the system should have secure authentication and authorization methods in place.

Techniques for appropriate encryption should be used when transferring and storing data.

To prevent unwanted access, the system should have strong access control and role-based permissions.

Usability:

The user interface ought to be simple, easy to use, and uniform on various platforms (web, mobile, etc.).

To assist users with a variety of tasks, the system must to offer precise and unambiguous instructions, error warnings, and help documentation.

Availability and Reliability:

To enable continuous scheduling and shift management activities, the system must have a high availability and little downtime.

To ensure system dependability, appropriate steps should be taken for load balancing, failover mechanisms, and fault tolerance.