|  |
| --- |
| SU-Logo-Black |

###### School of Computing and Artificial Intelligence

###### Faculty of Engineering and Technology

###### Sunway University

###### SYSTEM DESIGN SPECIFICATION

|  |  |
| --- | --- |
| **SEMESTER** | : JUNE 2025 |
| **COURSE NAME** | : BIS2102 INFORMATION SYSTEM ANALYSIS AND DESIGN |
| **LECTURER** | : ASSOC. PROF. TS. DR. ASLINA BAHARUM |
| **SYSTEM NAME** | : Smart Waitlist System for Enhancing iZone Subject Enrolment |
| **PROGRAM NAME** | : Bachelor of Information Systems (Honours) (Data Analytics) |
| **CONTACT PERSON** | : *Keertana, 23109614@imail.sunway.edu.my, 011-28353640* |

|  |  |  |  |
| --- | --- | --- | --- |
| **STUDENT ID** | **STUDENT NAME** | **GROUP** | **ROLE** |
| 23109614 | Keertana A/P Subramaniam | BIS 2-4 | Context Diagram, Data Flow Diagrams (Level 0, 1, and 2), Data Dictionary, Class Diagram |
| 22034540 | Wong Hui San | BIS 2-4 | Sequence Diagram and Use Cases Designer |
| 22053037 | Siow Qi Yung | BIS 2-4 | Use Case Designer and Class Diagram Contributor |
| 22064141 | Liu Yong Le | BIS 2-4 | State Transition Diagram Documentation |
| 22046015 | Azaliya Turlubayeva | BIS 2-4 | Sequence Diagram Documentation |
| 22017867 | Ayu Wen Li | BIS 2-4 | State Transition Diagram Designer |

|  |  |
| --- | --- |
| **DATE OF SUBMISSION** | 04/07/2025 |
| **ACTUAL DATE OF SUBMISSION** | 05/07/2025 |
| **NO OF DAYS LATE** |  |

**Table of Contents**

[1.0 Context Diagram 4](#_Toc203744452)

[2.0 Data Flow Diagram 5](#_Toc203744453)

[2.1 Data Flow Diagram Level 0 5](#_Toc203744454)

[2.2 Data Flow Diagram Level 1 7](#_Toc203744455)

[2.3 Data Flow Diagram Level 2 9](#_Toc203744456)

[3.0 Data Dictionary 11](#_Toc203744457)

[4.0 Use Case Diagram with Relationships 20](#_Toc203744458)

[5.0 Class Diagram 23](#_Toc203744459)

[6.0 Overall Sequence Diagram 31](#_Toc203744460)

[6.1 Use Case 1: Join Waitlist 33](#_Toc203744461)

[6.2 Use Case 2: Auto Enrollment with Notification 35](#_Toc203744462)

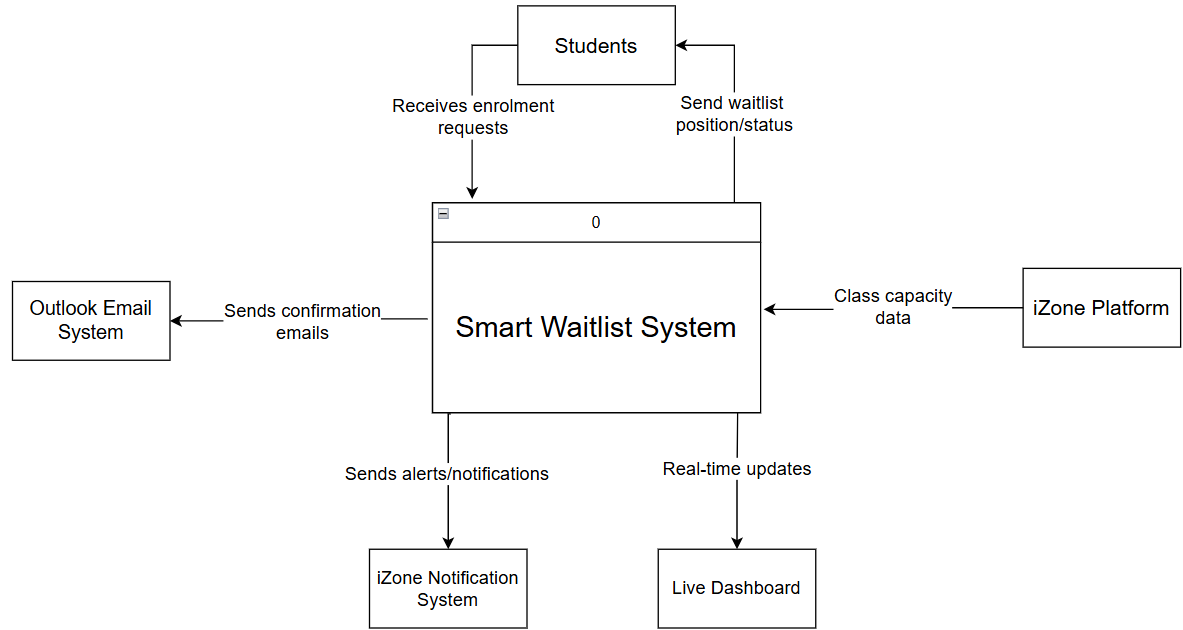
[6.3 Use Case 3: Live Dashboard 37](#_Toc203744463)

[7.0 State Transition Diagram 39](#_Toc203744464)

# 1.0 Context Diagram

The context diagram shows how the Smart Waitlist System interacts with Students, iZone Notification System, Live Dashboard System, Outlook Email System and the iZone Platform.

|  |  |
| --- | --- |
| **Entity** | **Role** |
| Smart Waitlist System | Handles all waitlist automation |
| Students | * Join waitlist * Cancel Waitlist * Views waitlist information on live dashboard * Receives alerts and email notifications |
| iZone Notification System | Sends real-time alert messages on the iZone homepage to notify students of any changes |
| Live Dashboard System | Displays waitlist information (position, number of students ahead, status) |
| Outlook Email System | Sends confirmation emails for any updates on waitlist |
| iZone Platform | Provides class capacity data and manages course enrolment and timetable data |



# 2.0 Data Flow Diagram

## 2.1 Data Flow Diagram Level 0

A diagram of a program

AI-generated content may be incorrect.

The diagram represents a high-level overview of the waitlist management system and its interaction with external entities. It includes one primary process: P1: Waitlist Management, which interacts with the Student and internal systems.

Key Features

* External Entity
  + Student: Can join, cancel, or view the waitlist
* Main Process:
  + P1: Waitlist Management – Handles student waitlist actions and communicates with the system for processing.
* Supporting Systems:
  + D1: Waitlist Database – Stores and manages waitlist data
  + P2: Auto Enrolment – Automatically checks and enrolls students based on class availability and schedule conflict.
  + D2: Class Timetable – Provides class slot availability
  + D3: User Profiles – Used to verify schedule conflicts
  + P3: Notification Manager – Sends updates through Outlook and iZone
  + Live Dashboard: Displays real-time waitlist updates.

## 2.2 Data Flow Diagram Level 1

A diagram of a program

AI-generated content may be incorrect.

This diagram breaks down the Waitlist Management process into sub-processes, detailing specific student actions and system interactions.

Sub-processes:

1. 1.1 View Classes:
   1. Students retrieve class info from D2: Class Timetable.
2. 1.2 Join Waitlist:
   1. Adds a student to D1: Waitlist Database
   2. Validates the entry using D3: User Profiles
   3. Updates 1.4 Waitlist, which displays real-time updates in the Live Dashboard
3. 1.3 Cancel Waitlist:
   1. Removes the student from D1: Waitlist Database
   2. Updates 1.4 Waitlist and Live Dashboard
4. 1.4 Waitlist:
   1. Central waitlist view
   2. Pushes real-time updates to Live Dashboard

## 2.3 Data Flow Diagram Level 2

A diagram of a flowchart

AI-generated content may be incorrect.

This diagram breaks down the Auto Enrolment process, specifically how the system selects and enrolls a student when a class slot becomes available.

Sub processes:

1. 2.1 Monitor Class Capacity:
   1. Monitors class availability using D2: Class Timetable.
2. 2.2 Select Next Waitlisted Student:
   1. Uses FCFS (First-Come-First-Serve) logic to pick the next student from the waitlist
3. 2.3 Check Conflict:
   1. Cross-reference students’ existing schedule from D3: User Profiles for any conflicts.
4. 2.4 Allocate or Skip & Reorder Waitlist
   1. If there is no conflict, the student is enrolled
   2. Updates:
      1. D1: waitlist Database (removes the student or reorders)
      2. D2: Class Timetable (decreases available capacity)
   3. Live Dashboard
5. 3.0 Notify Student:
   1. Sends email or alerts via Notification Manager

# 3.0 Data Dictionary

Table: Students

This table contains the personal and login information of students who uses iZone system where each student has a unique identifier and can access features like waitlists, receiving alerts, and viewing their live dashboard status during the enrolment period.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Data Type** | **Length** | **Constraints** | **Default Value** | **Relationships** |
| StudentID | Unique identifier for each student | Integer | 8 | Primary Key | N/A | Referenced in: WaitlistEntry, Notifications, LiveDashboard |
| FullName | Full name of the student | String | 100 | Not Null | N/A | N/A |
| Email | Student’s registered university email address | String | 100 | Not Null, Unique | N/A | N/A |
| IsLoggedIn | Student’s current login status | Boolean | N/A | Not Null | False | N/A |

Table: Subjects

This subject represents the academic courses offered at Sunway University depending on which semester. Each subject has a unique code and includes metadata such as subject name and its respective credit hours.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Data Type** | **Length** | **Constraints** | **Default Value** | **Relationships** |
| SubjectCode | Unique identifier for each subject (Ex: BIS1234) | String | 20 | Primary Key | N/A | Referenced in: TimeSlots, Waitlist |
| SubjectName | Name of the subject | String | 100 | Not Null | N/A | N/A |
| creditHours | Credit hours for the subject | Integer | 1 | Not Null | N/A | N/A |

Table: Timeslots

The timeslots table stores the scheduled class sessions such as tutorials, lectures, workshop for each subject. It includes the capacity and the number of students enrolled as well.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Data Type** | **Length** | **Constraints** | **Default Value** | **Relationships** |
| slotID | Unique ID for each timeslot | String | 20 | Primary Key | N/A | Referenced in: Waitlist, LiveDashboard |
| SubjectCode | Links to Subjects table | String | 50 | Foreign key | N/A | References Subjects (SubjectCode) |
| ClassType | Type of session (Lecture/Tutorial/Practical) | String | 20 | Not Null | N/A | N/A |
| Time | Scheduled time for the time slot | String | 50 | Not Null | N/A | N/A |
| Capacity | Maximum number of students allowed | Integer | 8 | Not Null | N/A | N/A |
| EnrolledCount | Number of students currently enrolled | Integer | 8 | Not Null | 0 | N/A |

Table: Waitlist

This waitlist acts as a controller for each time slot’s

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Data Type** | **Length** | **Constraints** | **Default Value** | **Relationship** |
| WaitlistID | Unique ID for the waitlist | Integer | 8 | Primary Key | Auto-Increment | N/A |
| SubjectCode | Foreign key linking to Subjects table | String | 20 | Not Null | N/A | References Subjects(SubjectCode) |
| SlotID | Foreign key linking to TimeSlots table | String | 20 | Not Null | N/A | References TimeSlots(TimeSlotID) |

Table: WaitlistEntry

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Data Type** | **Length** | **Constraints** | **Default Value** | **Relationships** |
| EntryID | Unique ID for waitlist entry | Integer | 8 | Primary Key | Auto-Increment | N/A |
| StudentID | Foreign key to Students table | Integer | 8 | Not Null | N/A | References Students (StudentID) |
| WaitlistID | Foreign key to Waitlist | Integer | 4 | Not Null | N/A | N/A |
| Position | Queue position in the waitlist | Integer | 8 | Not Null | N/A | N/A |
| JoinTimestamp | Date and time student joined | DateTime | N/A | Not Null | CURRENT\_TIMESTAMP | N/A |
| Status | Current status | Enum (Waiting, Enrolled, Removed) | N/A | Not Null | ‘Waiting’ | N/A |
| IsAutoEnrolled | Flag indicating if student is auto enrolled | Boolean | N/A | Not Null | False | N/A |

Table: Notifications

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Data Type** | **Length** | **Constraints** | **Default Value** | **Relationships** |
| NotificationID | Unique ID for the notification | Integer | 8 | Primary Key | Auto-Increment | N/A |
| StudentID | Foreign key linking to Students table | Integer | 8 | Not Null | N/A | References Students(StudentID) |
| Message | Notification message content | String | 500 | Not Null | N/A | N/A |
| SentTime | Timestamp of when the message was sent | DateTime | N/A | Not Null | CURRENT\_TIMESTAMP | N/A |

Table: AlertNotifications

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Data Type** | **Length** | **Constraints** | **Default Value** | **Relationships** |
| AlertID | Unique ID for alerts | String | 20 | Primary Key | N/A | N/A |
| StudentID | Foreign key to Students table | Integer | 8 | Not Null | N/A | References Students(StidentID) |
| Message | Alert content | String | 500 | Not Null | N/A | N/A |
| Timestamp | Time created | DateTime | N/A | Not Null | CURRENT\_TIMESTAMP | N/A |
| IsRead | If alert has been read | Boolean | N/A | Not Null | False | N/A |

Table: LiveBashboard

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Column Name** | **Description** | **Data Type** | **Length** | **Constraints** | **Default Value** | **Relationships** |
| SlotID | Foreign Key linking to TimeSlots table | Integer | 8 | Not Null | N/A | References TimeSlots (SlotID) |
| StudentID | Foreign Key linking to Students | Integer | 8 | Not Null | N/A | References Students (StudentID) |
| SubjectCode | Foreign Key linking to Subjects | String | 50 | Not Null | N/A | References Subjects (SubjectID) |
| Time | Scheduled time for the time slot | String | 50 | Not Null | N/A |  |
| Queue Number | The student’s current position in the waitlist queue | Integer | 8 | Not Null | N/A |  |
| EnrolledCount | Number of students currently enrolled | Integer | 8 | Not Null | N/A |  |
| WaitlistCount | Number of students currently on the waitlist | Integer | 8 | Not Null | N/A |  |
| IsSlotFull | Boolean indicator if the slot is full | Boolean | N/A | Not Null | False | Calculated |
| AutoEnrolmentInProgress | Indicates if auto-enrolment is processing | Boolean | N/A | Not Null | False | N/A |

Relationships

* One-to-One:
  + One Student can only join one WaitlistEntry
  + One TimeSlot can only have one Waitlist
* One-to-Many
  + One Subject can have many TimeSlots
  + One Student can receive many Notifications
  + One Student can receive many AlertNotification

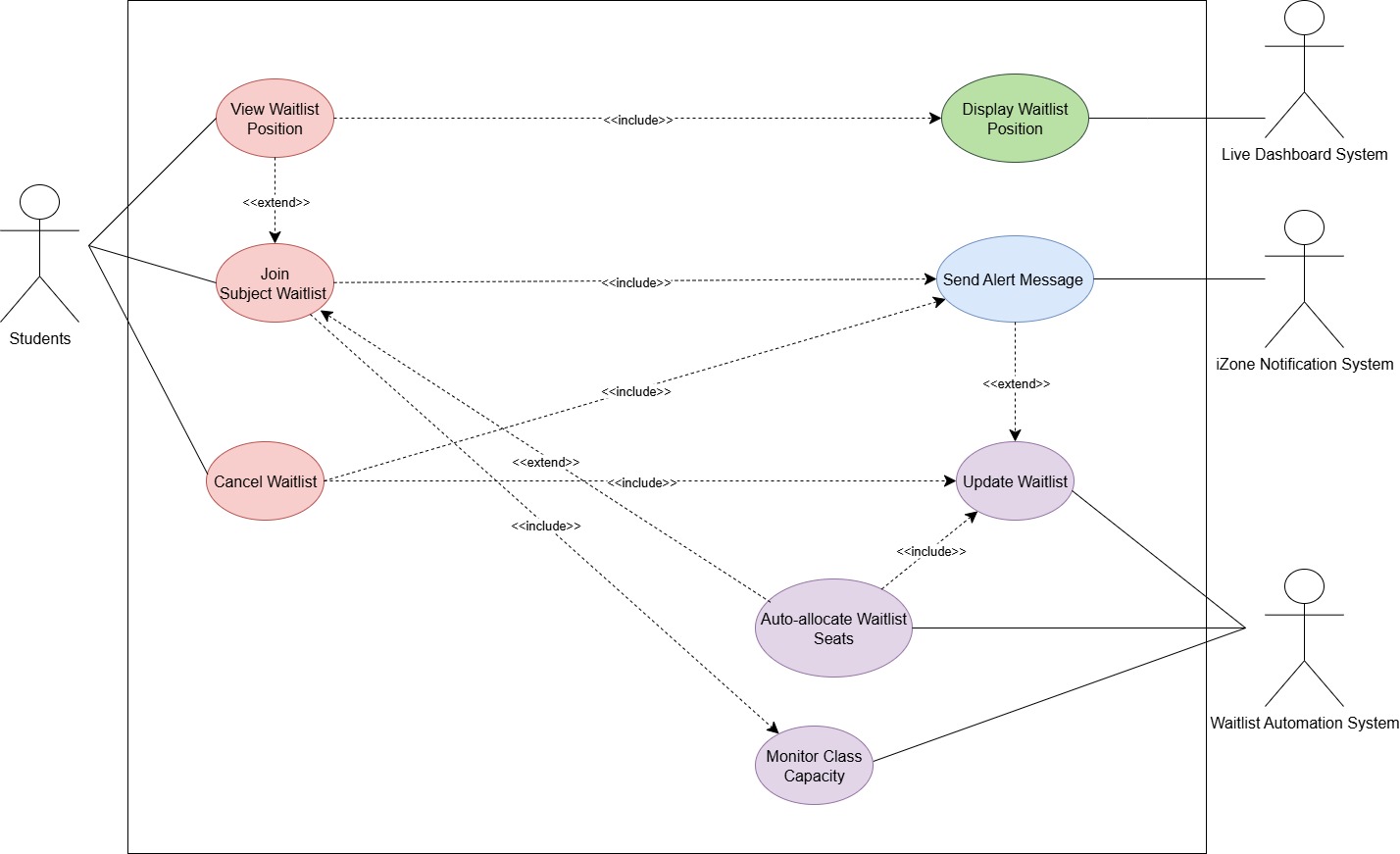
Metadata

* Data Source: iZone System Backend
* Owner: Sunway University’s IT Service Department
* Update Frequency: Real-Time (live updates during enrolment period)

Benefits of this Data Dictionary

* Clear understanding of how students, subjects, time slots, waitlists are structured
* Defines all necessary constraints to avoid duplicate waitlist entries per subject
* Supports auto-enrolment and notification workflows
* Provides a reference for system developers and university administrators

# 4.0 Use Case Diagram with Relationships



**Include & Extend Relationship:**

**Include Relationship**

1. Use Cases: **Join Subject Waitlist**, **Monitor Class Capacity**
   * The “Join Subject Waitlist” use case includes the functionality of the “Monitor Class Capacity” use case.
   * Waitlist automation system checks if a class is full before students joining the waitlist for a preferred time slot. Students can only join the waitlist after checking class capacity. If the class time slot is full, then students can choose to join a waitlist.
2. Use Cases: **Auto-allocate Waitlist Seats**, **Update Waitlist**
   * The “Auto-allocate Waitlist Seats” use case includes the functionality of the “Update Waitlist” use case.
   * Once a student is allocated a seat, the waitlist automation system must update the waitlist queue.
3. Use Cases: **Join Subject Waitlist**, **Send Alert Message**
   * The “Join Subject Waitlist” use case includes the functionality of the “Send Alert Message” use case.
   * Every time a student joins a waitlist, the iZone Notification System must send an alert on iZone homepage to alert the students to check their email for more details.
4. Use Cases: **Cancel Waitlist**, **Send Alert Message**
   * The “Cancel Waitlist” use case includes the functionality of the “Send Alert Message” use case.
   * When a student cancels a waitlist, the iZone Notification System must send an alert on iZone homepage to alert the students to check their email for more details.
5. Use Cases: **View Waitlist Position**, **Display Waitlist Position**
   * The “View Waitlist Position” use case includes the functionality of the “Display Waitlist Position” use case.
   * Once the student initiates the view action, the Live Dashboard System responds by displaying the relevant waitlist information such as position, and number of students ahead.
6. Use Cases: **Cancel Waitlist**, **Update Waitlist**
   * The “Cancel Waitlist” use case includes the functionality of the “Update Waitlist” use case.
   * After cancellation, the Waitlist Automation System must update the waitlist (e.g. remove the student, shift others up).

**Extend Relationship**

1. Use Cases: **Join Subject Waitlist,** **View Waitlist Position**
   * The “View Waitlist Position” use case may extend the “Join Subject Waitlist” use case.
   * Students can view their waitlist information (position, status and number of students ahead) only after they have joined the waitlist. Students must click Live Dashboard manually first to view their waitlist information.
2. Use Cases: **Join Subject Waitlist**, **Auto-allocate Waitlist Seats**
   * The “Auto-allocate Waitlist Seats” use case may extend the “Join Subject Waitlist” use case.
   * After joining a waitlist, the Waitlist Automation System will automatically allocate seats to students if a seat becomes available. Seat allocation only happens when a seat opens, the student is first in the queue and has no schedule conflict.
3. Use Cases: **Update Waitlist**, **Send Alert Message**
   * The “Send Alert Message” use case may extend the “Update Waitlist” use case.
   * After any updates from waitlist, the iZone Notification System sends alert messages on iZone homepage. Alert messages will pop up on the iZone homepage when a student joins a waitlist, cancels a waitlist, is auto enrolled in a class, has a schedule conflict and when the enrolment period ends.

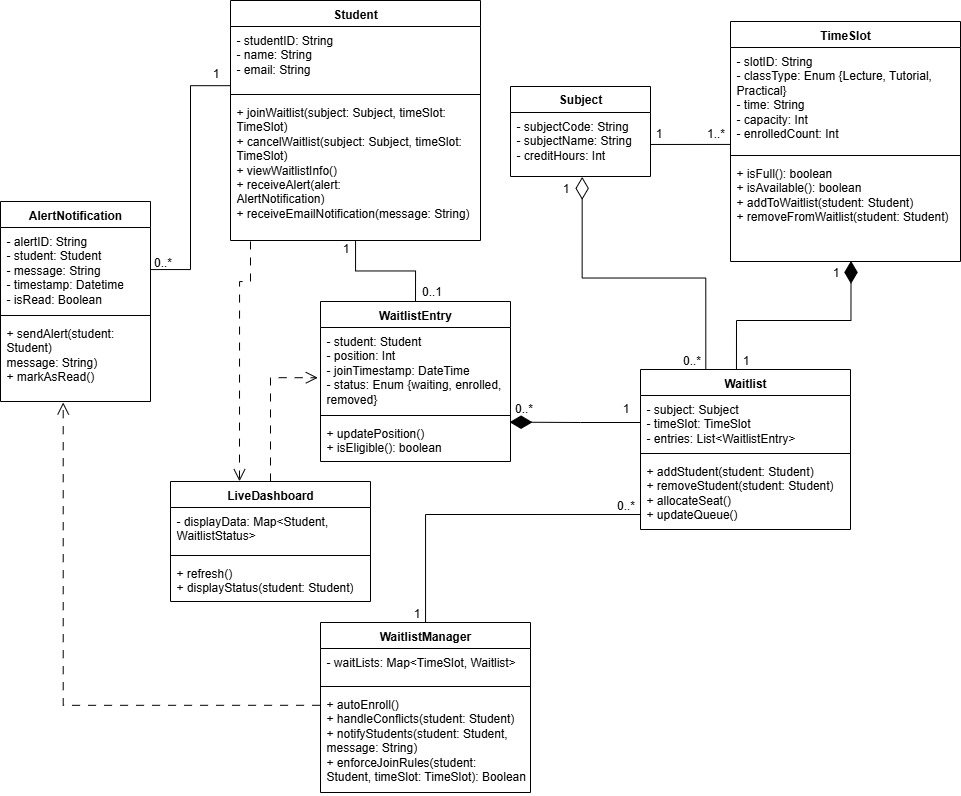
# 5.0 Class Diagram

In the Unified Modeling Language (UML), a class diagram is a kind of static structure diagram that illustrates a system’s classes, properties, operations, and relationships between its objects to explain its structure. An essential component of object-oriented modeling and design is class diagrams.

**Classes:**

* Each class is represented as a rectangle and is split into 3 parts:
* The top section is the class name
* The middle section is for the attributes
* The last bottom section is for the methods or functions the class perform

**Diagram:**



**Key Classes:**

**1. Students** - Represents a Sunway University student using iZone for subject enrollment

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Explanation** |
| studentID | String | A unique identifier of each student |
| name | String | Student’s full name |
| email | String | Student’s registered university email address |
| **Methods** | **Data Types** | **Explanation** |
| joinWaitlist(subject: Subject, timeSlot: TimeSlot) | - | Students join the waitlist for a specific timeslot |
| cancelWaitlist(subject: Subject, timeSlot: TimeSlot) | - | Cancels an existing waitlist entry |
| viewWaitlistInfo() | - | Opens the live dashboard to view the current position, status and number of students ahead |
| receiveAlert(alert: AlertNotification) | - | Receives visual alert (bell icon) on iZone homepage for waitlist updates |
| receiveEmailNotification(message: String) | - | Gets official email notifications via Outlook about waitlist actions |

**2. Subject** - Represents an academic subject offered by the university

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Explanation** |
| subjectCode | String | Unique code for each subject (e.g., BIS1234) |
| subjectName | String | The subject’s name |
| creditHours | Integer | Number of credit hours assigned to the subject |

**3. TimeSlot** - Represents a specific session under a subject

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Explanation** |
| slotID | String | Unique identifier for the time slot |
| classType | Enum {Lecture, Tutorial, Practical} | Type of class (Lecture, Tutorial, Practical) |
| time | String | Day and time of the class session (e.g., MON 8am-10am) |
| capacity | Integer | Maximum number of students allowed |
| enrolledCount | Integer | Number of students currently enrolled |
| **Methods** | **Data Types** | **Explanation** |
| isFull() | Boolean | Returns true if the number of enrolled students has reached capacity(enrolledCount ≥ Capacity) |
| isAvailable() | Boolean | Returns true if there are still available seats |
| addToWaitlist (student: Student) | - | Adds a student to the waitlist of this time slot |
| removeFromWaitlist(student: Student) | - | Removes a student from the waitlist |

**4. Waitlist** - Manages the list of students waiting for a full time slot

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Explanation** |
| subject | Subject | The subject the waitlist is linked to |
| timeSlot | TimeSlot | The specific time slot that’s full |
| entries | List<WaitlistEntry> | - All students waiting for this time slot  - A list of WaitlistEntry objects, ordered by join time |
| **Methods** | **Data Types** | **Explanation** |
| addStudent(student: Student) | - | Adds a student to the waitlist |
| removeStudent(student: Student) | - | Removes a student from the waitlist queue |
| allocateSeat() | - | Automatically enrolls the next eligible student if a seat opens |
| updateQueue() | - | Updates the queue positions after any change (e.g., a student cancels waitlist). |

**5. WaitlistEntry** - Stores individual student's information on the waitlist queue

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Explanation** |
| student: Student | Student | The student on the waitlist |
| position | Integer | Student’s current queue position |
| joinTimestamp | DateTime | Date and time the student joined the waitlist |
| status | Enum {waiting, enrolled, removed} | Current status (waiting, enrolled, removed) |
| **Methods** | **Data Types** | **Explanation** |
| updatePosition() | - | Recalculates the student's position when the queue changes |
| isEligible() | Boolean | Checks if the student can be auto-enrolled (no clashes, top of the queue, etc.) |

**6. WaitlistManager** - The backend controller for managing waitlists and automation

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Explanation** |
| waitLists | Map<TimeSlot, Waitlist> | This is a mapping of time slots to their respective waitlist queues |
| **Methods** | **Data Types** | **Explanation** |
| autoEnroll() | - | Automatically enrolls the next eligible student when a seat becomes available |
| handleConflicts() | - | Skip students from waitlist queue with class schedule conflicts |
| notifyStudent(student: Student, message: String) | - | Send an official email notification through Outlook about any changes or updates on the waitlist |
| enforceJoinRules(student: Student, timeSlot: TimeSlot) | Boolean | Prevents duplicate waitlist entries for the same subject |

**7. AlertNotification** - Handles the visual alerts on the iZone homepage (bell icon with red dot)

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Explanation** |
| alertID | String | Unique identifier for the alert |
| student | Student | The recipient of the alert |
| message | String | Alert message content |
| timestamp | DateTime | Date and time the alert was created |
| isRead | Boolean | Boolean value to mark if the alert has been seen |
| **Methods** | **Data Types** | **Explanation** |
| sendAlert(student: Student, message: String) | - | Sends a real-time visual alert to the student |
| markAsRead() | - | Marks the alert as read when the student views it |

**8. LiveDashboard** - A real-time student-facing dashboard to track waitlist status

|  |  |  |
| --- | --- | --- |
| **Attributes** | **Data Types** | **Explanation** |
| displayData | Map<Student, WaitlistStatus> | A mapping of students to their current waitlist status and position |
| **Methods** | **Data Types** | **Explanation** |
| refresh() | - | Refreshes dashboard data to reflect the most up-to-date waitlist status |
| displayInfo(student: Student) | - | Shows the student’s current position, number of students ahead, and status |

**Relationships:**

**1. Association** (solid line)

|  |  |  |
| --- | --- | --- |
| **Classes** | | **Explanation** |
| Student | WaitlistEntry | - A student can only have one waitlist entry across all time slots and subjects to simplify the system design and reduce the server load.  - Each WaitlistEntry is linked to one student.  - One-to-one relationships. |
| Subject | TimeSlot | - A subject can have many time slots and each subject should have at least one time slot for a class.  - One-to-many relationships. |
| WaitlistManager | Waitlist | - WaitlistManager manages multiple waitlists (One-to-many relationships). |
| AlertNotification | Student | - Each AlertNotification is sent to a specific Student.  - One student can receive many alert notifications (One-to-many relationships). |

**2.** **Aggregation** (hollow diamond)

|  |  |  |
| --- | --- | --- |
| **Classes** | | **Explanation** |
| Waitlist | Subject | - A waitlist is related to a subject, but the subject exists independently.  - A subject can have multiple Waitlists due to multiple time slots (One-to-many relationships). |

**3. Composition** (filled diamond)

|  |  |  |
| --- | --- | --- |
| **Classes** | | **Explanation** |
| Waitlist | WaitlistEntry | - Waitlist owns its entries. If the waitlist is deleted, entries will be deleted too.  - One-to-many relationships |
| Waitlist | TimeSlot | - Each waitlist is tied to a specific time slot.  - The waitlist cannot exist without its time slot.  - One-to-one relationship. |

**4. Dependency** (dashed arrow)

|  |  |  |
| --- | --- | --- |
| **Classes** | | **Explanation** |
| WaitlistManager | AlertNotification | - WaitlistManager depends on AlertNotification to send notifications. |
| LiveDashboard | WaitlistEntry | - The LiveDashboard depends on WaitlistEntry to display real-time waitlist information, but it does not own or modify the data, it only reads it for display purposes. |
| Student | LiveDashbaord | - Student depends on LiveDashboard to access or view waitlist information include their position, and the students ahead of them. |

# 6.0 Overall Sequence Diagram

A screenshot of a computer program

AI-generated content may be incorrect.

For a more detailed version which consists of every flows(main flows and alternative flows): [SequenceDiagram.org](https://sequencediagram.org/index.html?presentationMode=readOnly#initialData=AIZwLghgTmCuC2AbAUGAlmRBTABAHhDQC8sAuAZgAYA+AeQDcsoJFEcBlLAR1iwDsAxrgAiaCAHNm8ZBFhgA9hAHp6EdPL44NyGcvlQcAIgLEyAJhrs4AE35hDOCCA427yAA7R0AtJ75gjExJSC2o0AC0NXAAFfUhEByccWJgWDy80Hz8A40Jg0IB1CAxENHAcAEE5eXg1NA0OAE9wLHhE5yKSsrB0mEzfCH9AvPMaCKicADl5dAAzTLqG9mawVvapmbR5gUW+HQBeffwR0gBGAA5qaKgsAQ1rDHrNQ+QrWFshgFpqZLiWUmOpjOlwAMvJxGg9il4jhPt8XO87ACgmQLtQwRDNOxYAIhCAQDo3h8AvDof9AcE0QBRfxMHA0qDyRDwOy-VIoV6uL4-MmIZEnNEVMCreDuAIMplsSE4ABisFYHEQM2QvNh8KJSIpqMu7AAFvIAO5GABS8mlnUw3QcACE5Ao9pzEdy2fF+UC0QBhUoCADWOFN5uKlvAByOKOB1BYYAB7AEuqw1nluDufFm3oCcdufpeUY4mcT2BwKbTmQCWAAHt0CTgay6WLCfhbSuA3ZTLh7476cLN9Hn4wXcB6NCXlMhazgm901Ty-nytRGh6n07L5LA+NYx7XVfDpnMFuo+K3tdQACpQNDicR0xcj8VQRkGACyWHxEiwm5r2-VXOj87RhgDTQDSDZsAlmYpsGsbtewAbWxAAjAArW4AiHWwAF0ABocBggBxRlYHcKYEHgph0IAHXXXgcAUHAQHzJMi2HZcDQwXUcIQ5DlBwNCsCwnD8NXIjJhIsjDA-OiFHcZAsEQEBcGmOiGMLYt0wk7dGxA7ojwjCprCgk95AnLSQ3HSdyjhGd2R0tEACUsDgKBNAARV4ajYkIA8dHHDSNj3HYDxsy4zwvK8DBvNAoFqA9KmwGAcGfV8rwk3ctn3J5pwRYkguoQwAE1V0ATAJGBwUj+EcfSExooywHjHBgK6cAADpKOibAnGTTs-UaVcDFaCDoIMWxIDQOSmvE8d1NnBs6zncM0VEEAIHgwtAOMxqAltYVtH4DdkEOP9LijAEOyzHAtkkp0AjKRxEBuCBrEaLQgJMgIc0QAJJloE9nvq16pvZGbzN-eb2y6nAAElZli+7HpwWhNGBiTgcy3kct3eHEf+gGYXhYGcr0gyjKRszXtR2ccvsxyXLcmJ5E8p5vK3aad02bZdhykLL2vYdIuijKKjigJEqW5Lx1S9mYssrLNVB3KCtgYrcDKzQHtsKDaNq3AGuDMAWr4NqsA6otwZ62A+tqUbBpwYaIJAcamc-FmrNdQ7qEW5bVrNLGNpwLb7R0WT5JloYIfYW7YaehoddAnH6zx16ctOrsoZho24YR9bdeRsnpbRt3BcjzGs9jnznb8tKAqeTnz25gxhEI701FwQWmGFl9RffOO2GljV-By-LVxwXUIBKlgi8Q72qq1v6Nv1w3jfoNAsCNM2DBjqd3Hpx4GgaGeQTQErhCcXV4MUKBrAd3bQzd46mhaeAcCYB8bfNyFxFn3WcBuCFwGYGL3oBGWA-J+95ezyFxObaszNAYJw2gTfSOBDIl26DnX2ecKZuypGAqA3cZoS3ShoGuoU6TYJfq3eKIs3yO1mplPuIMBSXEMDKCC08jKT2lDPDezVKLJHasHMAUAnoSGKJoRAzcoBX3XDoJQCgDC5CBKEeh6x6G9G8AMIYCj8g0BRlUBQ-MlgrDWI4Do-1PB9CyIMHI4ZQjjD4ApNmhCsRGLaCYiuktGb7TDIwq4NwUwPAAfsR0xIgaJwLnddOT0OH2KgtwgIAAKWYjJH4AFVg4emNqcAAlDfOWd96FFkGEIOSn9QLf24LwcogCQ4ZiKUHUpU4bg8BfAEUiPYbh0SNtdZwo8IKe3fOOApcDdZJzqWwFG9lmmmVrCjaW+M3b2XgPIEqMpkkoJDGgr+rN-IczdlzMKPExk8V5lFXYsU24JQ7tQ8Wjiq4NF7j+AeCslZ0RxHiEAsx5SICeircpSzGBQSSTUGidU4nWxgiAWASEUL8RgugFkdElRgHQvrPhRtg6Zi7GvJ+ls2DtJtg5)

The above is the overall sequence diagram that shows only the main flows to provide an overall picture of the iZone Smart Waitlist System. It consists of the flow of the three main uses cases, from student joining waitlist to how the waitlist automation system will respond to student’s request meanwhile students are able to receive notifications in order to catch up with the latest updates. There is also a live dashboard which allows students to check their latest status and position in the waitlist. The overall sequence diagrams mainly uses loop and alternatives logic in fragments to cover all the conditional paths such as handling errors, cancellations, conflict scheduling and sending reminders.

## A close-up of a document AI-generated content may be incorrect.6.1 Use Case 1: Join Waitlist

For clearer view: [Join Waitlist Sequence Diagram.org](https://sequencediagram.org/index.html?presentationMode=readOnly#initialData=FAAQzgLghgThCuBbANsCBLCyCmACAPGOgF7YBcAzAAwB8AqmHgMJSO4CMZuAUgPboA7XAGV4AIwBW2AMYRcAdSiZk6SMCjwIvKLPQA3KBl5DjwdbN4xcAIkIlyAJlrCEAE2wCI13KxFuPEMAADrAY0ughnjZ2pGRONOgAWsZ4AAqW0MjevulwUKghcOjhkXK2RLHxisqqcgCCmryIhujGIgCekNiI2WAKSli1waHFEVBR5fZxtEkpuAByvBgAZsUtbcKdEN29C0voq9LrAmYAvKcEFeTsABw0qTAyxq6YrULnwC7w7lEAtDS4XKZLgxa53AAyvAA5oJgED8rhfv8-N8AiCrmRbjRITChKJpNJsGAwGYvj85Mj4ch0VMsQBRTzYKwMmC8ZCIAKAjL5Myk-x-AFUmmxLF1CDbRBBOQstnIXCCXAAMXgyDlwmQSzh3LlSIBZLRl1pd2EAAteAB3Gx8BXVQaQbwAIU0WhOn35FMF2uFYJoTBU0gA1jx+EJbSo1MBzoaRXd8hAuMJpCbsK4VXhpMZlv65EmZEGPnGRLnUzhcBmBFninJsAAPWok3CNrl5HUAsO1b2Yu5MZOB3DLSxF5Ml5iZ7PAJv9GqQRGUr3Rn1MMdVpW8eACVwTptUxEAxYrNZGASdrEAFRg6ChUKZuCXFezuDpMFZVgAskSwFBr1vGzvdSjyRPO5rGtIRzQGcM5GWJQcFcftBwAbVESQZDkJd3AAXQAGlwBCAHFWXgIIFiQMQmQwgAdDd4DwLRcDAYs0zLZdZFwc1MBNXDkKkVj0OwbDcIItdiPmUjyOsH96K0IJgGwZA2EWejGNLctK1kSS-zbCCOwXLsaDqVw4NPXgpztQJJ3bGd-yFXSsQAJWwBAYCEABFGiaK5IgjzMSdNL2A8jiPICaHPS9rysO9VhgZoj1wOocDgXB32JL9sEk-cDkPN5Zz1d1gusABNNdAEwCPQ8DIjwfEMlNcDoiBkzY7TIAAOio1IcFYdNeyDdo1ysboYPgqx3GgdB5OaiTJw07Vd2bYFbLuAARVQoDEUtQNMyDcCdcVTA8TdIwuUE9LjLgezzeVlik1EolUHxkEeKBXHaXA2nA6c5ALZA5HmAB5U9XrAprzO3GbkUs+MFt9brcAASSu+LHue3BftDYHJIhnK5vyYL9xRtGPp80GW1miHgoMoyTIhjHgaxmzjvsxz4Gc3A3OwDz0i8t4id-MG932Q5jmC0KrxvSL0Gi444oSuRks-b9JwywXYv-fVPHyor4FK8rsEqp73DguqGveszWoEdrsE6ssYd65ncAGsahtwEaYLACaeex1tPeC5bPzWvANsxnaXTMOS2DVuRYeEe6kZet70d8vnNp0hnuxh+Hpdj-Hk7UCzaes+dU-0h7LeR1Gc5B3mSeRJWsuMYWL1FqxFqI-1DDweKmVlj9Uo9v9kQjjW11wE0oDKmPS5eiQQxq+q8BNyCzYtq29HQbBLVtqwF9qXAgl4Lm2jaOfcHBfQ8EW1gTTEbQYFcd39rOI6MSxU6Oi6RB7efQdUwvAQoUaj6uBHgwkgDAKWX05CbHfp-F8r0CTMwbMTTIpNgbk0MrgYyFcaaAILi2YKT4XzTWrvzAKQsoYi3Co+L+VhO6JTlr3IhyDVZ5ShtYRUMFZ4mWngqY+28WpUUBB1NgEAYAvS-EoIQyB24wHvhuXkIB9pIFQEAA)

The sequence diagram depicts a student’s workflow for joining a subject waitlist in the iZone Portal. The actors involved are the students, iZone Portal, Waitlist Automation System and iZone Notification System. It starts with the student logging in, navigating to the enrollment portal, and trying to enroll in a particular subject. If the subject is fully booked, the system shows a button “Join Waitlist”. After students click ‘Join Waitlist’, the system will first check whether the time slot that students intended to join waitlist clashes with their current enrolled time slots. If there is a schedule conflict, an error message will be prompted. Otherwise, the system will proceed with verifying whether the student is on the waitlist for that subject. In case the student is not on the waitlist, they are added with a queue position returned and confirmation alert triggered. Otherwise, alert message is displayed. There is also an alert for errors in registration attempting to process the requests. These notifications are sent to the students by the iZone Notification System.

## A screenshot of a computer screen AI-generated content may be incorrect.6.2 Use Case 2: Auto Enrollment with Notification

For clearer view: [Auto Enrollment Sequence Diagram.org](https://sequencediagram.org/index.html?presentationMode=readOnly#initialData=-QkkHiDUp4AdAAdEa7bBIkkbrQAaxwpR7JxKmyRAd0cHB-EgBUQEfZxkCkXwMTbN5qB4fghFEcRJCSfY+FHKYLmQe8+CiB4FAVXApDECRiSidNrFBWVTAsLBkFDW57nYmMDRxRNTD-dicAARSZJluGEUEvCwaUuHlRVhQVDBSnRNlQmkkVDlXZk5RApUVXVANHUbTjdXAbj23eYwMOo0DlCwAAPdlNUiAzBKbEzzXMlCPS9XARBskENXCYk-VXQz6nRVzTI42tI3rfQvLjRMUzTJw0IEfzxCLdMZHLMcq1YgyIzmYZG2bZzWzdHZnEymikiHEcKwnUDdyvQ9jzMC8rxvSV7zPF83w-VgtG-P8ANgICigVMCnEgkCYMi9UW2MoLrkSnEI1HYJixkSR8rTYIJK-Q4fymYCFXReCJKuxCkKq3F8U4WrsJkOikz4UEEIIpZDHxaysNo+iPpwL6azcjzA3WnYAElrlmX4PWyYQgWTcL8hbEAweilauOQuN0u4McUxGWTdOVXB4zQW8vD6qYLVaXxV2nLHt3ctHKtjd1Kl8wnCGTYFJIqqZQrhdjFpDbHwxmErozx3ik1TdMMNBAB5dR-rqmQPhyssKwKmtiqjBs2ZbKHqBqzDNZwN6GJwBQQgQkEBu7PsB1nEYWD5gWcHQvcvFgI8Ty6vdKepvqn1fRhRv-QCQamlrwK0RUxwgrAoMpWCbvRYAEIQFAgA)

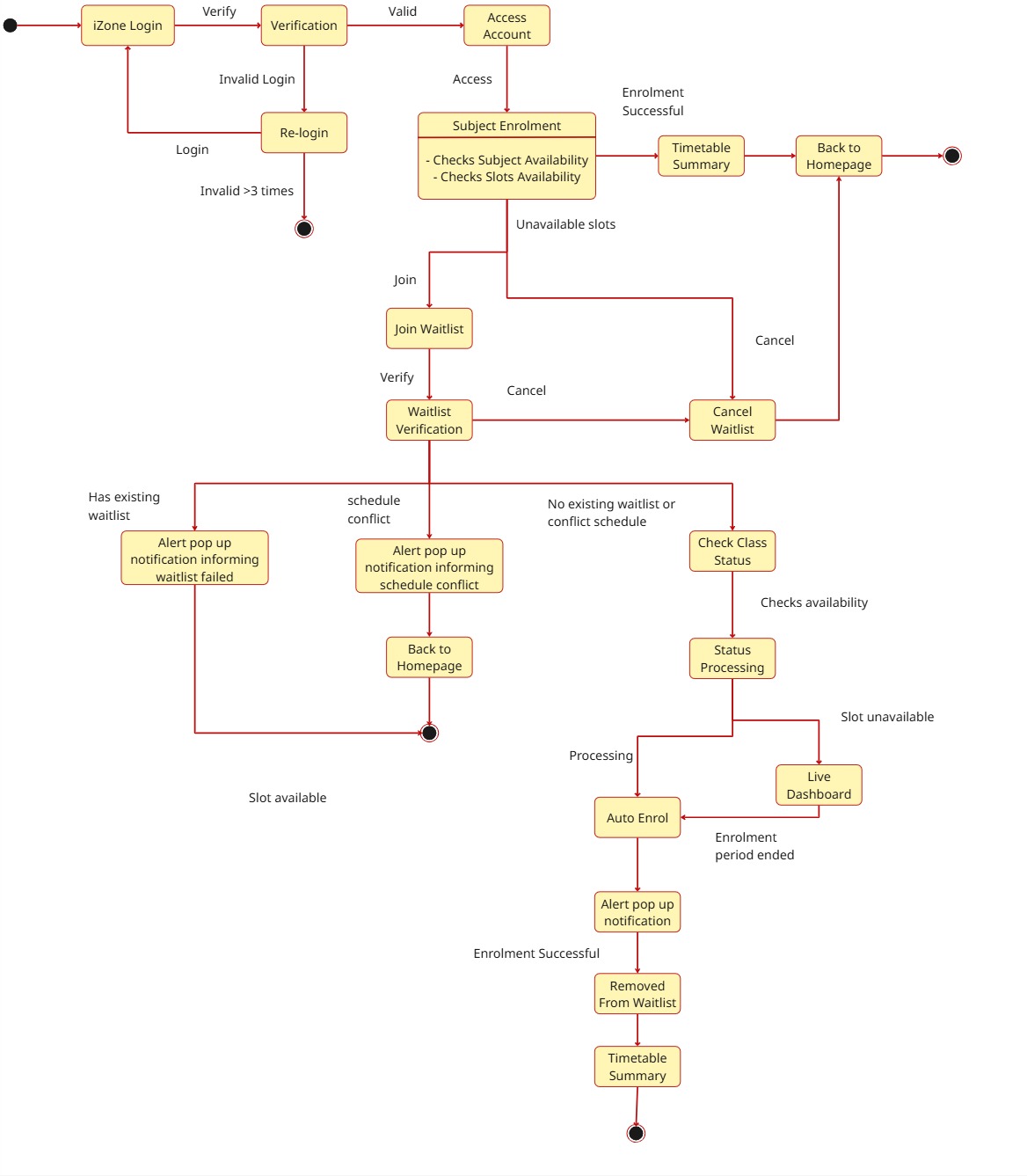
The sequence diagram above shows the use case of Auto Enrolment with Notification for students who are on a subject waitlist. The actors involved are the students, Waitlist Automation System and iZone Notification System. To begin, the process starts with the Waitlist Automation System monitoring the queue to look for available seats. Whenever a seat opens, the system automatically enrolls the next student in line, drops them from the waitlist, and sends out an auto-enrolment alert via iZone Notification System. This cycle will loop while the enrollment period remains active. If the enrollment period ends before all students are automatically enrolled, the system then randomly assigns the remaining slots to waitlisted students, disenrolls them from the waitlist, and sends an end-of-enrollment notification. Moreover, students can at any time decide to cancel their waitlist request, triggering the system to remove them from the list and send cancellation confirmation.

## A screenshot of a computer screen AI-generated content may be incorrect.6.3 Use Case 3: Live Dashboard

For clearer view: [Live Dashboard Sequence Diagram.org](https://sequencediagram.org/index.html?presentationMode=readOnly#initialData=-Y3k6Uz1LIF4dKO8oTl+05MrO8TsguPpLhC1B8iAApCjguiwFAVSVumIHOAqvboiksi6IKlxXgosHMvYwb0fBrKIfORLtKhwwAEpYFCnToKq9BbjuVzMc4hKWHxHRiNUEhCZ8aiCmI2DiTqCEGshkYwtGZRGEwtRjFA2KCiBIAAISHsegCYBJEJZltgsggKc5yXHIsCIIgwoKEoADaJxSjKsISLYAC6rq+QA4qWsB5m8CCyVAoUfmpzIaUhupZE8Jo8tQvHwBIkTgNp8ilvAr7kWA4k-owqVMuxOZeGIegItVdRZkyjEdTqyA1WqPqzLiEGPlM8rCmRDJVeprGaRx2xcXpR4ntAuDDU+L4TbaPkkcKzmBfoIApV1yB5Dly7tlwACidiluWtxVjWIXFnUuItO2ODXbdFYdA9tbPdIVzTXqqwZYuZ1oUYqhviRsj2XdlbVn9pJ2XUe4ADp2Ol9YXlaVEOnun5A1ls0aQt6GaBAKm4ICmX6jgABCsBgAYdi9e1yDAIxCAoEAA)

Th sequence diagram above demonstrates Use Case 3 which is Live Waitlist Dashboard detailing the student’s engagement flow with the Live Dashboard System for tracking their waitlist. The actors involved are the student and Live Dashboard System. The process starts with logging into iZone, during which the live dashboard is accessed, and the student dashboard loads. For students entered a waitlist, the system shows pertinent subjects and time slots with the student's position, status, and number of students ahead. Throughout the enrollment period, the dashboard offers real-time updates. Data is refreshed dynamically including changes to the student’s position or status. Alert message will also be displayed on dashboard to students who is enrolled automatically when seat becomes available, at the same time, it will trigger removal of enrolled students from the waitlist. In cases where the student has no active waitlists, the system presents corresponding notifications. Upon the enrollment period ending, the live dashboard button will be disabled and students will not be able to view the live dashboard.

# 7.0 State Transition Diagram



|  |  |
| --- | --- |
| **State** | **Explanation** |
| Log In into iZone | This is the first state where the users attempt to access the iZone portal. They will be required to input their credentials such as their id and password to begin the authentication process. After they had entered their credentials, the system will move to the verification state to check if the credentials are valid. |
| Verification | In this state, the system verifies the users’ login credentials. If the login is valid, the user will granted access to their account. If the login is invalid, the system will move the user to a re-login state. This validation process ensures only authorized users gain access to sensitive enrolment functionalities. |
| Re-Login | If the verification is invalid, the student will be moved to this state. They will be prompted to try logging in again. If there are repeated invalid attempts (e.g., more than three times), the system may block access or require additional authentication. Otherwise, they are allowed to retry the login process. |
| Access Account | Once the student’s login is valid, they will move into this state, where they can access the full features of their account. From here, they can navigate to subject enrolment actions such as joining or cancelling a waitlist. It serves as the base from which students manage their academic registration activities. |
| Subject Enrolment | This is a decision-making state after accessing subject enrolment. The student may choose to join a waitlist or cancel their request. This state leads the user to the appropriate pages based on their action, ensuring that the correct next steps are performed. |
| Cancel Waitlist | When a user chooses to cancel joining the waitlist, the system will move them here. After the waitlist request is cancelled, the user will be brought back to the home page. |
| Join Waitlist | If a user chooses to join a waitlist for a full class, the system will move them into this state. The system records their request, assigns them a position in the queue (based on first-come-first-served), and displays their current position. |
| Waitlist verification | This state verifies whether the user has an existing waitlist or has a scheduling conflict with an already enrolled subject. |
| Homepage | This is the main iZone homepage where the user can navigate to the subject enrolment page, the live dashboard and view the notifications. |
| Alert pop up notification informing waitlist failed | If the user already has an existing waitlist enrolled, they will be unable to join another waitlist due to the rule of one waitlist per student. User will receive a pop-up notification informing them of this issue. |
| Alert pop up notification informing schedule conflict | If the waitlist slot that the user wishes to join clashes with a subject slot that they have already enrolled for, u ser will receive a pop-up notification informing them of this issue. |
| Check Class Status | This state represents the system checking for seat availability. The system will continuously monitor whether seats open in the desired class. This happens behind the scenes in real time and drives the next state based on availability and eligibility. |
| Status Processing | If the class is not full, this state evaluates if the user is eligible for enrolment (e.g., no timetable clash, prerequisites met). If they are not eligible, the system loops them back to waitlist monitoring. If eligible, they will proceed to automatic enrolment. |
| Live Dashboard | User can view the position of their queue while they are waiting for a slot to open up. |
| Auto Enrol | In this state, the system attempts to automatically register the student into the desired class. This only happens if the student is at the top of the waitlist, a seat is available, and no conflicts exist. It’s fully automated and runs without manual intervention. |
| Alert Pop Up Notification | The system will send a pop-up notification on iZone to alert the user that once a slot opens and the student has enrolled in the desired class and a successful enrolment email has been sent to the user’s email. |
| Send Notification | The system will automatically send the user an email via Outlook to let them know that they have been enrolled in the class they chosen. |
| Removed from Waitlist | Once the student is successfully enrolled, they are removed from the waitlist queue to prevent duplicate allocations. This is a cleanup state that ensures queue accuracy and system integrity. |
| Timetable Summary | After a student is successfully enrolled in their preferred class (either through auto-enrolment or direct access), the system moves to the Timetable Summary state. In this state, the student can view their updated class schedule in a consolidated format. |