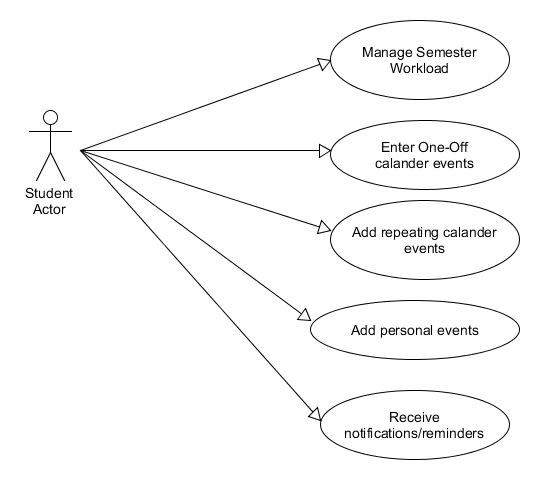
# Use cases

## Use Case - Model 1



## Fully Dressed Use Case - Model 1

**Primary actor:** Student user

**Goal in context:** User wishes to manage workload, enter one off and repeating and personal calendar events and receive notifications from the software.

**Level:** User Level

**Stakeholders and Interests:**

User: Wants to keep up to date with college events and workload

**Preconditions:** System is correctly keeping track of time and date and awaiting inputs from the user

**Minimum Guarantee:** Failure message stating event could not be created.

**Success Guarantees:** User created event is stored within the calendar and a timer has been started for notification upon completion

**Trigger:** User starts interaction by opening software and logging in.

**Main Success Scenario:**

1. User opens digital diary software
2. User proceeds to login successfully to the system
3. User selects that they wish to create an event
4. System prepares itself for event creation
5. User adds time, date etc. To the event creation
6. System notifies user of event creation
7. System adds event to calendar and begins timer countdown

**Extensions:**

1. User enters incorrect login details. Proceeding is held until correct info is entered
2. User creates and clashing event with event already in the system

**Frequency of Use:** Several times per day

**Priority:** 1

## Use Case - Model 2

## 

## Fully Dressed Use Case - Model 2

**Primary actor:** Lecturer user

**Goal in context:** User wishes to enter one off and repeating and personal calendar and public events, module information and receive notifications from the software.

**Level:** User Level

**Stakeholders and Interests:**

User: Wants to manage module info for students, add public events for students of a certain module and add personal events

**Preconditions:** System is working sufficiently and is awaiting input to display data

**Minimum Guarantee:** Failure message stating event could not be created or module info could not be added.

**Success Guarantees:** User created event is stored within the calendar and a timer has been started for notification upon completion

**Trigger:** User starts interaction by opening software and logging in.

**Main Success Scenario 1:**

1. User opens digital diary software
2. User proceeds to login successfully to the system
3. User selects that they wish to create an event
4. System prepares itself for event creation
5. User adds time, date etc. To the event creation
6. System notifies user of event creation
7. System adds event to calendar and begins timer countdown

**Main Success Scenario 2:**

1. User opens digital diary software
2. User proceeds to login successfully to the system
3. User selects that they wish manage module information
4. System prepares itself
5. User adds info to the module of their choice
6. System notifies user successful changes made

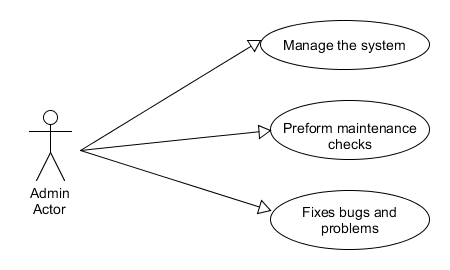
**Extensions:**

1. User enters incorrect login details. Proceeding is held until correct info is entered
2. User creates and clashing event with event already in the system
3. User tries to change module info that they do not have access to

**Frequency of Use:** Several times per day

**Priority:** 1

## Use Case - Model 3



## Fully Dressed Use Case - Model 3

**Primary actor:** Admin user

**Goal in context:** User wishes to maintain the software.

**Level:** User Level

**Stakeholders and Interests:**

User: Wants to maintain the software for reliability for use by students and lecturers throughout the semester

**Preconditions:** System is waiting for admin inputs

**Minimum Guarantee:** Failure message stating that system is not functioning properly and needs maintenance

**Success Guarantees:** Admin preforms tests on the system that return successful

**Trigger:** User starts interaction by opening software and logging in.

**Main Success Scenario 1:**

1. User opens digital diary software
2. User proceeds to login successfully to the system
3. User selects that they wish to perform maintenance tests on the software
4. System prepares itself for maintenance and removes itself from server until finished
5. User preforms tests
6. System notifies user of test results

**Extensions:**

1. User enters incorrect login details. Proceeding is held until correct info is entered

**Frequency of Use:** Several times per day

**Priority:** 1