CIS 350 02

WINTER 2022

Term Project – Release 1

Ben Allen allebenj@mail.gvsu.edu

Devin Elenbaase elenbaad@mail.gvsu.edu

Bryan VanDyke vandybry@mail.gvsu.edu

Studious

A student organizer.

<PRETTY SCREENSHOT HERE>

Contents

[Studious: A student organizer. 3](#_Toc94605751)

[Project Description 3](#_Toc94605752)

[Use Case 4](#_Toc94605753)

[Use Case Diagrams 4](#_Toc94605754)

[Use Case Descriptions 4](#_Toc94605755)

[Design Diagrams 5](#_Toc94605756)

[Source Code Anal 6](#_Toc94605757)

[Coding Standards 6](#_Toc94605758)

[Bug Checks 6](#_Toc94605759)

[Code Repository 7](#_Toc94605760)

[Repository 7](#_Toc94605761)

[Project Website 7](#_Toc94605762)

[Testing 8](#_Toc94605763)

[Unit Tests 8](#_Toc94605764)

[Code coverage 8](#_Toc94605765)

[Member Roles 9](#_Toc94605766)

[Ben Allen 9](#_Toc94605767)

[Devin Elenbaase 9](#_Toc94605768)

[Bryan VanDyke 9](#_Toc94605769)

[Self-Reflections 10](#_Toc94605770)

[Ben Allen 10](#_Toc94605771)

[Devin Elenbaase 10](#_Toc94605772)

[Bryan VanDyke 10](#_Toc94605773)

[Project Demo 11](#_Toc94605774)

**Green text is from the rubric. TODO: ~~mark done~~ as needed (or delete)**

# Studious: A student organizer.

## Project Description

Project description & List of features implemented in release 1

Sample screenshots of your application 10

Studious is planned to be an android-based student organizational assistant, with primary feature goals.

1. Calendar/Event-planner allows students to keep track of their upcoming exams, projects, presentations, etc., in one place.
2. Ability to track how many work hours have been committed to a single project/goal.
3. Ability to set weekly goals/to-do lists for workhours put toward specific projects.

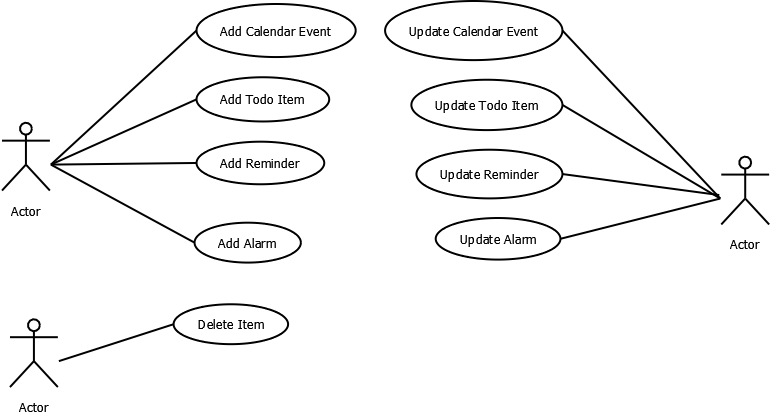
These features will assist a student in organizing and using their time efficiently through the chaos that is college scheduling.

A possible secondary stretch feature would be a repository for students to upload a particular professor's general assignment scheduling for a specific class. For example, it would allow students to see if their CIS 350 class will have a semester-long group project as opposed to CIS 241 having four individual projects throughout the semester.

## Use Case

### Use Case Diagrams

Use case diagram (system boundary diagram)



### Use Case Descriptions

Use case descriptions (using the template provided on Blackboard) 20

|  |  |
| --- | --- |
| **Name** | Add Calendar Event |
| **ID** | UC1 |
| **Brief Description** | Add Calendar Event |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  . . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

|  |  |
| --- | --- |
| **Name** | Add Todo Item |
| **ID** | UC2 |
| **Brief Description** | Add Todo Item |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  .  .  . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

|  |  |
| --- | --- |
| **Name** | Add Reminder |
| **ID** | UC3 |
| **Brief Description** | Add Reminder |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  .  .  . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

|  |  |
| --- | --- |
| **Name** | Add Alarm |
| **ID** | UC4 |
| **Brief Description** | Add Alarm |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  .  .  . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

|  |  |
| --- | --- |
| **Name** | Update Calendar Event |
| **ID** | UC5 |
| **Brief Description** | Update Calendar Event |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  .  .  . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

|  |  |
| --- | --- |
| **Name** | Update Todo Item |
| **ID** | UC6 |
| **Brief Description** | Update Todo Item |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  .  .  . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

|  |  |
| --- | --- |
| **Name** | Update Reminder |
| **ID** | UC7 |
| **Brief Description** | Update Reminder |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  .  .  . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

|  |  |
| --- | --- |
| **Name** | Update Alarm |
| **ID** | UC8 |
| **Brief Description** | Update Alarm |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  .  .  . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

|  |  |
| --- | --- |
| **Name** | Delete Item |
| **ID** | UC9 |
| **Brief Description** | Delete Item |
| **Actors (primary and supporting/secondary)** | Primary (use case initiator) and supporting actor(s) |
| **Triggers** | Event that gets the use case started |
| **Preconditions** | Conditions that must be true before the use case can start.  Written in the present tense. |
| **Primary Flow** | Main success scenario (aka primary or happy scenario) as a sequence of actions steps by various actors and system.  - Each action step is written to show a simple, active action  - Number the steps  Example:  1.  2.  3.  4.  .  . |
| **Alternate Flows** | For every point (step) in the basic flow where the behavior can branch because of a particular condition  - Write down the condition  - Write the steps that handle it  - Number the steps within each condition  Example:  2a condition that causes a branch in the main flow  2a.1 action step 1  2a.2 action step 2  .  .  .  . |
| **Minimal Guarantees** | A minimal guarantee represents a condition that will be true when the use case ends, regardless of how it terminates.  Written as assertion(s) in the past tense. |
| **Success Guarantees** | A success guarantee represents a condition that will be true when the use case ends successfully, regardless of which path it took.  Written as assertion(s) in the past tense. |

## Design Diagrams

Design diagrams (such as Class diagrams) 15

## Source Code Anal

Usage of static source code analyzers for

### Coding Standards

• Enforcing coding standards/conformance (using tools like Checkstyle or other IDE/language-specific)

### Bug Checks

• Finding potential bugs in the source code (using tools like SpotBugs or other IDE/language specific) 10

## Code Repository

### Repository

~~URL to application code repository (on GitHub)~~

<https://github.com/AllenStudent/CIS-350-Term-Project.git>

git@github.com:AllenStudent/CIS-350-Term-Project.git

### Project Website

~~Project website on GitHub Pages at http://username.github.io/repository/ 10~~

<https://github.com/AllenStudent/CIS-350-Term-Project>

## Testing

Unit tests and code coverage reports from unit testing and functional/system testing using tools such as

• JUnit (and EclEmma for Eclipse)

• Or, other appropriate tools for IDE/language used 15

### Unit Tests

### Code coverage

## Member Roles

Roles/Responsibilities of each team member of the project

### Ben Allen

### Devin Elenbaase

### Bryan VanDyke

## Self-Reflections

Self-reflection by each team member 10

### Ben Allen

### Devin Elenbaase

### Bryan VanDyke

## Project Demo

Project Demo (not part of the release document; TBD) 10