Group Design Document

Daniel Banks - 100093180
Danny Murphy - 100082602
Kyle Alexander - 100082709
Patryk Ogrodnik - 100051155

Link to trello board: https://trello.com/b/vpzDUAtk

Table Of Contents

Systems Analysis

Evernote Analysis - www.evernote.com

Pros

- Available on mobile
  - Including widget features
- Free version
- Online and desktop application
- Multi platform - desktop, web, mobile, etc.

Cons

- Simplistic UI makes some features difficult to discover
- Some features locked behind premium - $45
  - Data cap - 60MB for free, 1GB for premium
  - Offline use
  - Share notes
- Overcomplicated
- Feature Creep - Bloated with useless features

Features

Create Notes
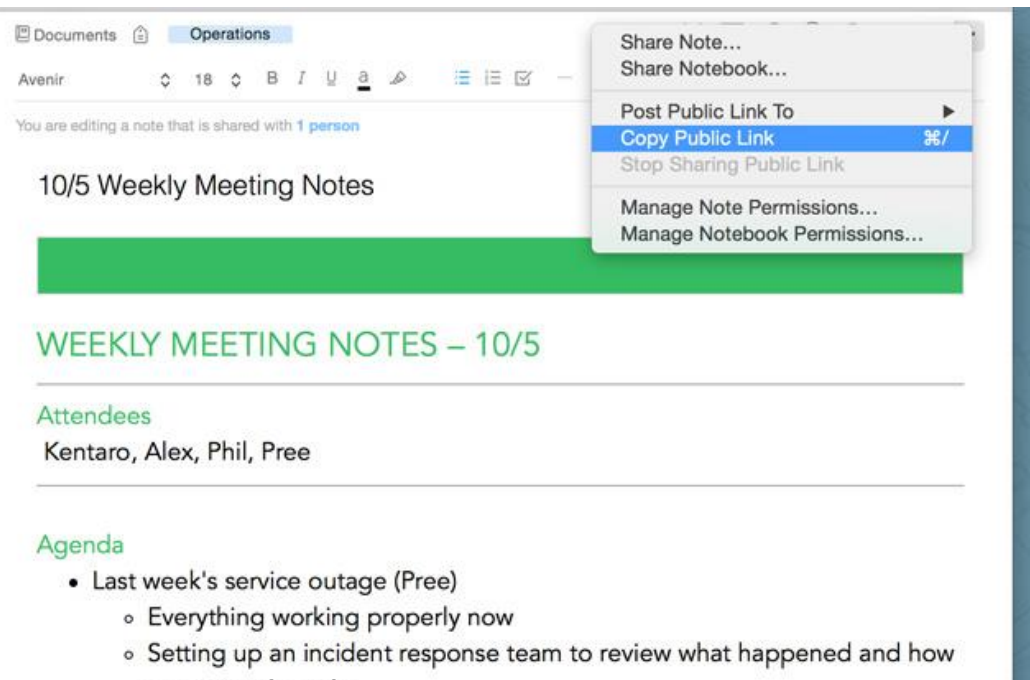

Search Notes

Organise Notebook
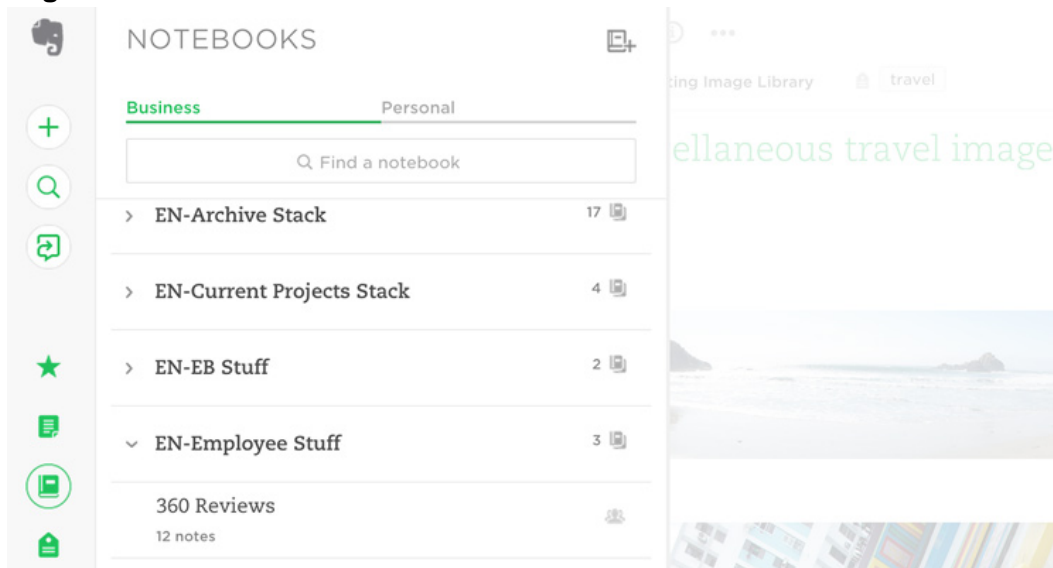
Share Notes

Layout
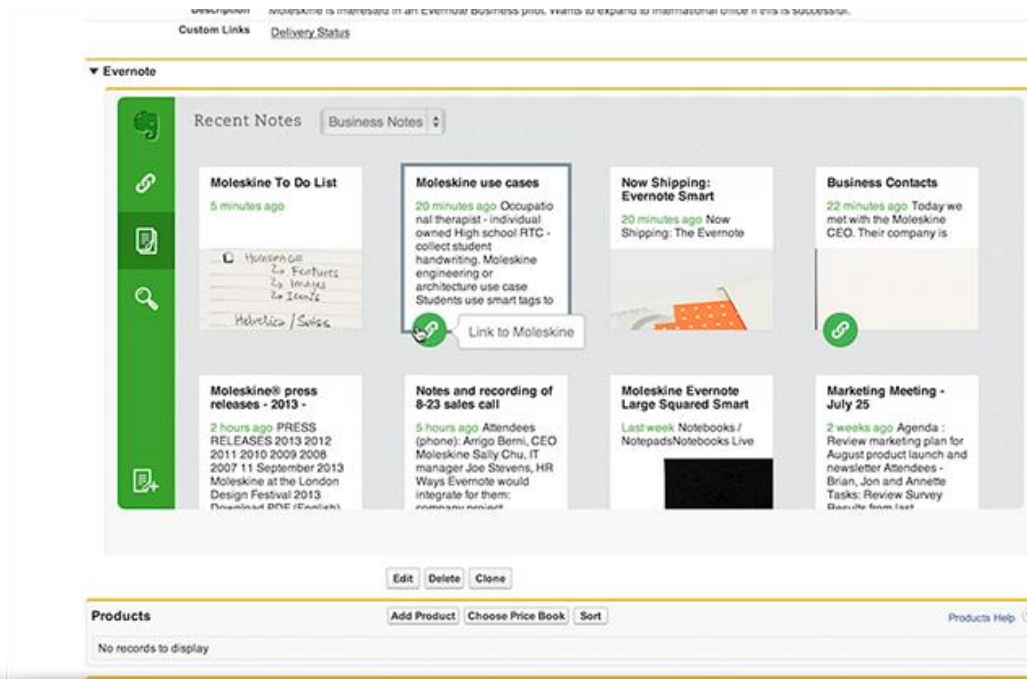
Goconqr Analysis - www.goconqr.com

Pros

- Has the ability to personalise the user's experience based on what stage of education they are at (e.g primary, secondary, undergrad etc.)
- Easy to create flash cards
- Free associated mobile apps
- Ability to build mind maps
- Limited Free version
- Easy to setup
- Works offline
- 2 million online learning resources
- Integration with social media accounts (Facebook, Google+)
- Inter-user communication
- Built-in planner and calendar

Cons

- Some feature nice features are locked behind premium include - £10
  - Ad free
  - Unlimited tasks
  - Unlimited public resources
  - Private resources
- No Desktop Application - Web based only

Features

Mind maps
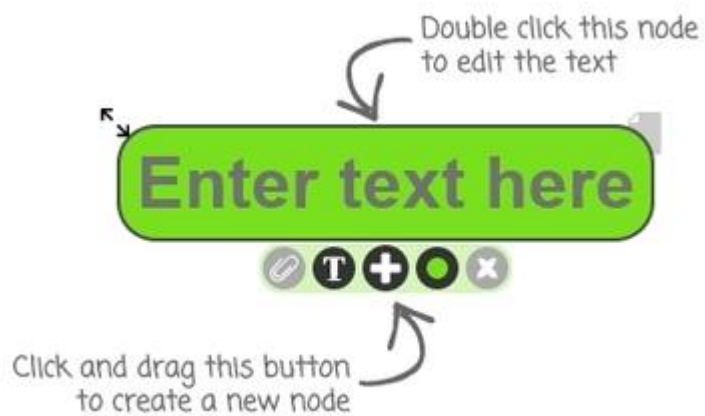
FlashCards
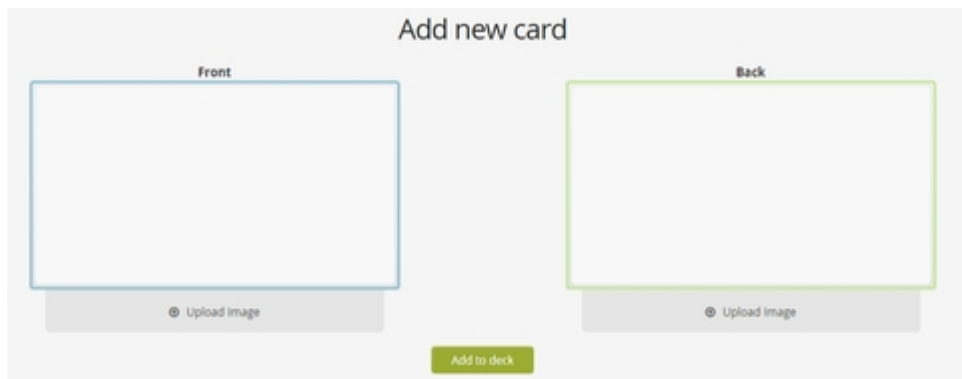
Notes Taking
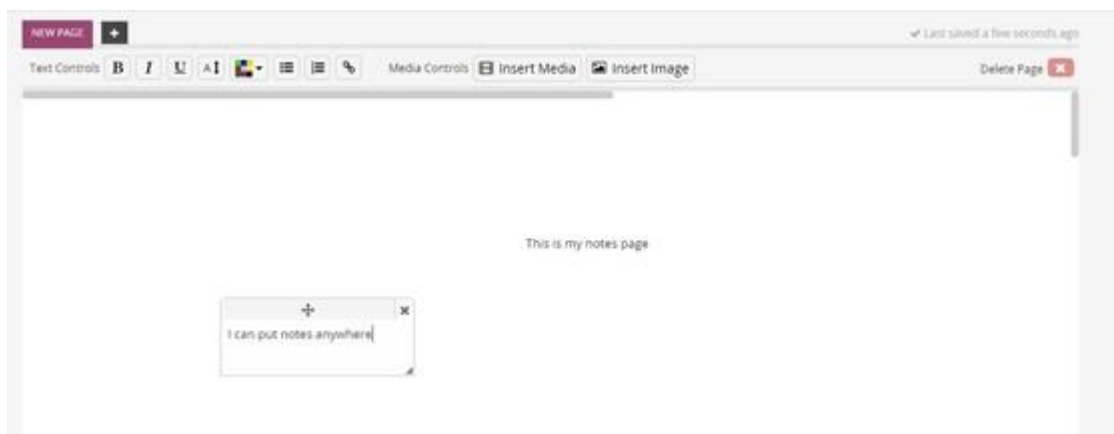
Personalise Experience

MyStudyLife Analysis - www.mystudylife.com

Pros

- Multi-platform - Desktop, mobile
- Free version
- Cloud storage
- Personalized view of your daily activities
- Allows you to supervise the progress of long term activities
- Ad free
- Assign relationships between tasks and classes
- task manager to remind yourself of tasks
-

Cons

- No reminders
- Doesn't allow users to add classes, tasks and exams from the calendar page
- Only designed for classes, nothing else

Features

Timetable

Glossary

Flash Card - a card that has words, numbers, or pictures on it and that is used to help students learn about a subject.

Example: https://algebra-regentsprep.wikispaces.com/file/view/Flashcard_Template.jpg/218331740/Flashcard_Template.jpg

Premium Features - features of a piece of software that the user must pay either a subscription fee or a one off payment to access (e.g £9.99 for unlimited storage space)

Mind Map - a diagram used to visually organize information. A mind map is often created around a single concept, drawn as an image in the center of a blank landscape page, to which associated representations of ideas such as images, words and parts of words are added.

Example: http://www.mindmapping.com/img-content/mind-map.jpg

Feature Creep - is the ongoing expansion or addition of new features in a product. This can often lead to a lot of unwanted features which ultimately wastes development time and money.

Cloud Storage - is a model of data storage in which the digital data is stored in logical pools, the physical storage spans multiple servers (and often locations), and the physical environment is typically owned and managed by a hosting company.

OMT - object modelling technique. A way to look through our requirements specifications and using nouns as classes and verbs as relations

OOA - object oriented analysis. using the result of our OMT and building a class diagram from the problem domain. In this diagram we only present classes, relations, fields and methods relevant to the user and the requirements

OOD - object oriented design. This is a diagram built on top of an OOA diagram and includes classes from the solution domain. this might include database and UI classes as well as toolkit classes like math libraries

Lo - Fi - a low fidelity prototype is a low quality mock-up of what a UI might look like

MoSCoW - is a way of categorizing a list of requirements for a system into Must have which are core features of the system, Should have, Could have and Won't have which are all features that would benefit the system but have to sometimes be cut out due to time/money constraints.

UML - Short for Unified Modelling Language. is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system.

Use Case - is a way in which the users will interact with the system

Use Case Diagram - A simple diagram which shows an overview of all the use cases each actor can perform within the system

Actor - Anyone that interacts with the final system

System - The system refers to the overall piece of software

GUI - Short for Graphical User Interface. An example of a GUI would be a form that a user has to interact with.

Requirements MoSCoW Final

Must
- Be able to load a semester file containing module, coursework and deadline information, and exam information from a defined file format
- Be able to define study milestones which must be attached to coursework or Exams
- Be able to define study tasks contributing towards specific coursework or exams
- Be able to store defined milestones and tasks on disk as part of the semester profile

Should
- Be able to record study activities that are related to study tasks and contribute towards completing milestones
- Be able to store activities in the semester profile file
- Have a study progress dashboard that highlights upcoming deadlines, progress towards completing milestones and time spent for each module
- Be able to visualise activities, dependencies, milestones and deadlines in a Gantt chart representation

Could
- Be able to process updated semester files with new deadlines and updating the stored activities, tasks and deadlines
- Be able to highlight the critical path of activities within the Gantt chart
- Be able to visually highlight progress made in the Gantt chart based on study activities

Wont
- Support direct communication with the hub via internet
- Download semester profile files from an external resource
- Require hub or faculty staff members to interact with the system directly

This was updated to better fit the client's needs and to remove a lot of the previous Musts which were not as core as we first thought. This final requirements MoSCoW was agreed upon by the client and was used to prioritize tasks when developing the application.

Use cases

USE CASE NAME	New Semester Profile	
Goal in Context	Loads in semester data from a file provided by the HUB	
Scope & Level	Overall System	
Preconditions	Program is running and the user is on the Overview Dashboard	
Success End Condition	A new profile is created and the user is prompted to locate a file containing the semester information. The data is loaded into the application	
Failed End Condition	Program fails to load a semester profile. An error message is displayed to the user stating that the file load failed, and why it failed, e.g. empty file selected, etc.	
Primary Actor	Student	
Trigger	The User wants to open a New Semester Profile.	
SUCCESS SCENARIO	**Step**	**Action**
	1	User presses the "New Semester Profile" button on the Overview Dashboard
	2	File Explorer prompt is generated asking user for file location
	3	User selects semester file given by the HUB
	4	Semester information is loaded into the application
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	3a	User selects a file that is NOT from the HUB
	4a	Error occurs and no information is loaded in, displaying an error message to the user
RELATED INFORMATION		
Priority	Must have.	
Performance Target	Needs to return either success or failure within 5 seconds	
Frequency	Used twice per academic year	
SCHEDULE	Version 1.0 release	
AUTHOR	Kyle Alexander 100082709	

USE CASE NAME	Add milestone
Goal in Context	Create a milestone and attach it to a semester profile

Scope & Level	Overall System
Preconditions	A semester profile must have been created/loaded in. A task must exist within the assessment the milestone is being attached to. The user is on the Module Overview window.
Success End Condition	A milestone is attached to an assessment
Failed End Condition	A milestone is not created
Primary Actor	Student
Trigger	The user wishes to add a milestone to one of their assessments

SUCCESS SCENARIO	**Step**	**Action**
	1	User presses the "add milestone" button
	2	User fills out name, description and deadline and selected related task(s) from a list
	3	System attaches milestone to the assessment and it is added to the milestone list on the module window
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	2a	User fills out form with incorrect data.
	2b	All fields are validated on focus lost, so any incorrect fields are outlined in red.
	2c	The user presses the submit button when there are errors, an error message is displayed and the submit will fail until all errors are corrected.

RELATED INFORMATION	
Priority	Should Have
Performance Target	Milestones should be added in 5 seconds or less after the actor submits
Frequency	Frequent
Channel to Primary Actor	User Interface
SCHEDULE	Version 1.0 release
AUTHOR	Kyle Alexander 100082709

USE CASE NAME	View gantt Chart
Goal in Context	Display a Gantt chart to the user

Scope & Level	Overall System
Preconditions	A Semester Profile must be loaded in. The user is on the Module Overview Window for the module they want to see the gantt chart for.
Success End Condition	A Gantt chart is displayed to the user
Failed End Condition	The program displays an error saying the information is corrupt
Primary Actor	Student
Trigger	The user wants to view a gantt chart detailing Assessments, tasks, activities and milestones for the chosen module.

SUCCESS SCENARIO	Step	Action
	1	User presses the "view gantt chart" button
	2	A new window opens to show a gantt chart containing module information

ALTERNATIVE SCENARIO	Step	Branching Action
	2a	An error occurs when opening the new window.
	2b	Display an error message to the user detailing what the issue is.

RELATED INFORMATION	
Priority	Could have
Performance Target	From button press to display is 5 seconds
Frequency	Very frequently
Channel to Primary Actor	User Interface
SCHEDULE	Version 1.0 release
AUTHOR	Kyle Alexander 100082709

USE CASE NAME	Add Activity
Goal in Context	To allow the actor to create and attach/link an activity to an existing task
Scope & Level	Overall system
Preconditions	The system is running and there is an existing task that has been loaded into the program.

	THe user is on the Module Overview window.
Success End Condition	A new activity is created and is attached/linked to the task selected by the actor and can be viewed in detail
Failed End Condition	A new activity is not created and the actor is prompted with an error message and is given the chance to try again
Primary Actor	Student
Trigger	The user wishes to add an activity to an existing task.

SUCCESS SCENARIO	**Step**	**Action**
	1	The user presses the "Add Activity" button on the module window
	2	The system displays a form requesting details for a new study activity consisting of a name, description, start date and end date to the actor
	3	The user enters the activity details into the system, selects an assessment and selects a task to attach to
	4	User presses the "submit" button
	5	All fields validate correctly
	6	The system creates the new activity and updates the selected task to now contain the newly created activity
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	3a	All fields are validated on focusLost. The system finds an error with the information entered by the user and outlines the incorrect fields in red.
	4a	The user presses submit with errors in the form.
	4b	An error message is displayed to the user, asking them to correct the errors
	4c	The submit fails until all errors are corrected

RELATED INFORMATION	
Priority	Should have
Performance Target	The activity should be created and attached to a task within 5 seconds of the user submitting the form successfully.
Frequency	Very frequently
Channel to Primary Actor	User Interface
SCHEDULE	Version 1.0 release

AUTHOR	Danny Murphy, 25/01/2016

USE CASE NAME	Select Module
Goal in Context	To select correct module and it's relevant information
Scope & Level	Overall System
Preconditions	System is running and displays area with module list. A semester profile is loaded in. User is on the Overview Dashboard or a Module Overview page with a nav bar.
Success End Condition	The user is able to select module from his course/semester
Failed End Condition	An error occurs when selecting a module, displaying an error message to the user detailing the problem.
Primary Actor	User
Trigger	User wants to select one of their modules to view it in more detail.

SUCCESS SCENARIO	Step	Action
	1	The User selects a module from the list on overview dashboard or the nav bar
	2	The system displays an area with modules from the current semester taken by the user
	3	Actor selects module from the area
	4	System displays a module window that contains relevant information about selected module
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	3a	System is unable to display module page due to insufficient permission
	4	Error is displayed to the user and system goes back to select page

RELATED INFORMATION	
Priority	Top priority
Frequency	Frequent
Channel to Primary Actor	User interface
OPEN ISSUES	What happens if system displays modules which are not taken by the actor?
SCHEDULE	Release 1.0
AUTHOR	Patryk Ogrodnik, 24/01/2016

USE CASE NAME	Open Semester Profile	
Goal in Context	To load content of module profiles with corresponding course resources	
Scope & Level	Overall system	
Preconditions	System is running and has valid resources data loaded	
Success End Condition	User is able to view information about the module: assignments, exams, weightings and deadlines	
Failed End Condition	System fails to load the file	
Primary Actor	User	
Trigger	User wants to open an existing semester profile	
SUCCESS SCENARIO	**Step**	**Action**
	1	User presses the "open semester profile" button
	2	System displays a file explorer window
	3	User selects the file they want to load into the system
	4	User presses on the select file button
	5	File is successfully loaded into the system
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	4a	User select cancel to close the window (file is not being loaded)
	5a	File fails to load
	5b	System displays an error message to the user detailing the error, e.g. "selected file is empty".
	5c	System closes the File explorer.
RELATED INFORMATION		
Priority	Top priority	
Performance Target	The file should be uploaded and the data should be updated in 10 seconds	
Frequency	Frequent	
OPEN ISSUES	What happens when system doesn't load the file? What happens when some of the loaded files are missing?	
SCHEDULE	Release 1.0	
AUTHOR	Patryk Ogrodnik 24/01/2016	

USE CASE NAME	Add note to task
Goal in Context	To add a note to a created task
Scope & Level	System scope, User Level Use Case

Preconditions	There is a Task already created to add a note too.
Success End Condition	Task note created
Failed End Condition	Task note not created
Primary Actor	User
Trigger	The user wants to add a note to a task

SUCCESS SCENARIO	Step	Action
	1	User selects a study task
	2	The user presses the "add note" button
	3	The user enters a note description
	4	The user presses the "Submit" button
	5	Note is successfully added to Task

ALTERNATIVE SCENARIO	Step	Branching Action
	3a	User leaves the note description empty
	3b	User presses the submit button
	3c	System displays an error message stating that a description is required

RELATED INFORMATION	
Priority	1
Performance Target	Task note is added instantly
Frequency	Frequent
SCHEDULE	Version 1.0 release
AUTHOR	Daniel Banks

USE CASE NAME	Save file
Goal in Context	To save a user's semester profile to disk
Scope & Level	Overall system
Preconditions	System is running and has valid resources data loaded
Success End Condition	All semester data including tasks, activities and milestones created by the user are saved to their local disk
Failed End Condition	System fails save the file
Primary Actor	User
Trigger	User wishes to save the date in their current semester profile to open at a later date
SUCCESS SCENARIO	Step	Action

	1	User presses "save" button
	2	"Save as" window opens up prompting user to provide a filename and choose a location
	3	User presses save
	4	Data is saved to a file in the location the user has selected with the given file name
	5	File is successfully saved to the local disk
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	2a	If the user had previously opened a saved file then the file is overwritten and the user is not presented with a file chooser
	4a	If the user does not provide a name then it is saved as "newSemesterProfile" by default
	5a	File is not saved successfully as an error has occurred
RELATED INFORMATION		
Priority	Top priority	
Performance Target	The file should saved without any corrupt data	
Frequency	Very Frequent	
OPEN ISSUES		
SCHEDULE	Release 1.0	
AUTHOR	Danny Murphy 27/04/2016	

USE CASE NAME	Add Task	
Goal in Context	To create a study task	
Scope & Level	System scope, User Level Use Case	
Preconditions	The system is running and the user is on the Module Overview window	
Success End Condition	Study tasks created	
Failed End Condition	Study task not created	
Primary Actor	User	
Trigger	User wants to add a Task to an assessment.	
SUCCESS SCENARIO	**Step**	**Action**
	1	User presses "Create task" button
	2	User enters - task name, date, associated module, task description, associated assessment event, requirements criteria & task dependencies.

	3	User presses the "submit" button
	4	System validates all input successfully
	5	Study task is created
ALTERNATIVE SCENARIO	**Step**	**Branching Action**
	4a	Error with entered details.
	4b	System displays an error message that states what errors occurred and incorrect fields are outlined in red.
RELATED INFORMATION		
Priority	1	
Performance Target	Study task is created without any errors	
Frequency	Frequent	
Channel to Primary Actor	User Interface	
SCHEDULE	Version 1.0 release	
AUTHOR	Daniel Banks	

USE CASE NAME	Add note to an activity	
Goal in Context	To allow an actor to add a text-based note to an existing activity	
Scope & Level	Overall System	
Preconditions	There must already be an existing activity that has been attached to an existing task	
Success End Condition	A note is successfully added to a given activity and can now be accessed by the user	
Failed End Condition	A note is not made and the actor is prompted that an error has occurred and is given the chance to try again.	
Primary Actor	Student	
Trigger	The user wants to add a note to an activity.	
SUCCESS SCENARIO	**Step**	**Action**
	1.	The user selects an existing activity and presses on the "Add a note" button.
	2.	The user enters the note details into the system
	3.	The systems validates the information that was entered
	4.	The system creates a new note, attaches it to the selected activity and displays it to the user

ALTERNATIVE SCENARIO	Step	Branching Action
	3a	The system finds an error while trying to validate the new note and displays an error message to the user
	4a	The system does not create the activity and instead redisplays the form to the user
RELATED INFORMATION		
Priority	Should have	
Performance Target	The note should be created and update the activity within 5 seconds of the user submitting the form successfully.	
Frequency	I expect this to be used a moderate amount.	
Channel to Primary Actor	User Interface	
SCHEDULE	Version 1.0 release	
AUTHOR	Danny Murphy, 25/01/2016	

USE CASE NAME	Update Deadline	
Goal in Context	To update one or more assessment deadline(s)	
Scope & Level	Overall system	
Preconditions	System is running and an existing semester profile has been loaded	
Success End Condition	All assessment deadlines are up to date	
Failed End Condition	The assessment deadlines are not updated and the original semester profile remains the same	
Primary Actor	Student	
Trigger	The HUB sends the user a new semester file with updated deadlines	
SUCCESS SCENARIO	Step	Action
	1.	The user presses the "Update Semester Profile" button
		The system displays the file explorer
	2.	The user navigates the file explorer and selects the new semester profile file given by the HUB
	3.	The user confirms the new file upload
	4.	The system loads in the selected file and overwrites the current file

ALTERNATIVE SCENARIO	Step	Branching Action
	3a	The system finds an error while trying to load the file and displays an error message to the actor. The semester file remains the same
RELATED INFORMATION		
Priority	Should have	
Performance Target	The update should happen within 3 seconds of the user submitting the form successfully	
Frequency	I do not expect this to be used a lot as deadlines are not often changed unless a student asks for an extension.	
Channel to Primary Actor	User Interface	
SCHEDULE	Version 1.0 release	
AUTHOR	Danny Murphy, 21/01/2016	

Use Case Diagram

This use case diagram is an overview of how the user will interact with the system. It contains several key actions the user will be performing which were described fully in the textual use cases.

Use Case Diagram
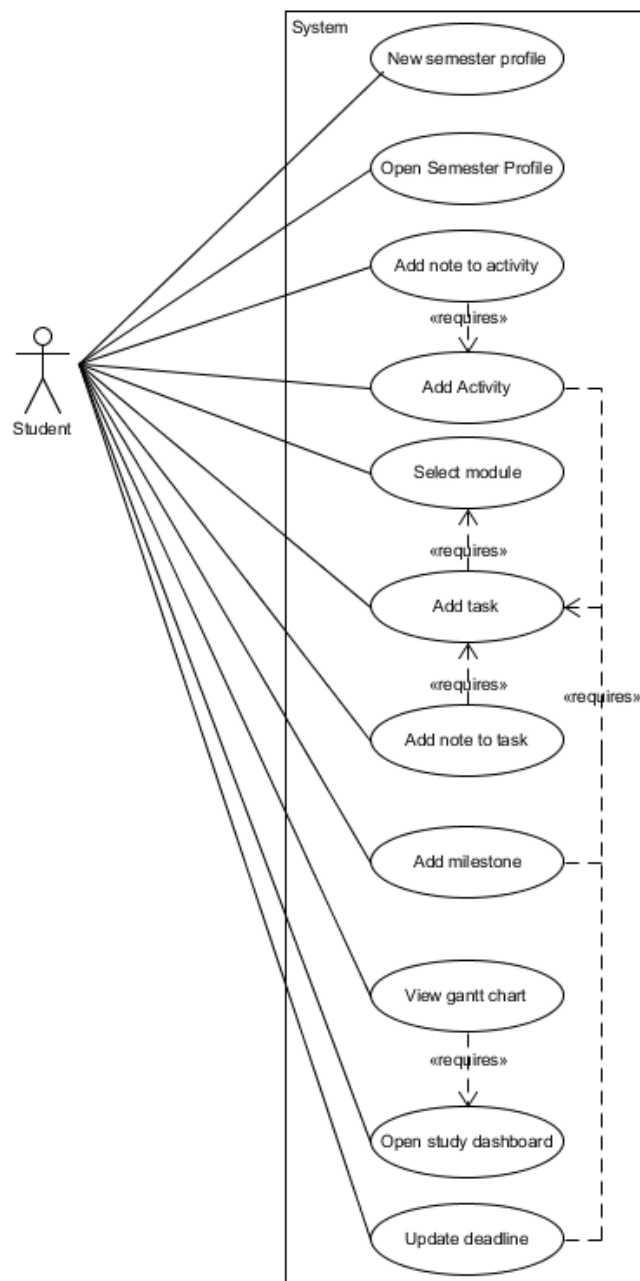
Changes

At a point in our implementation we spoke to the client and they said that updated deadlines are simply going to be given out as new files by the hub to the student/user via an email and will have to be uploaded by them. This meant that module organiser was no longer an actor in our use case diagram and was removed from our design.

Changes to use cases

A few things were changed/updated with our use cases. Firstly the way in which we Create a new semester file has changed slightly. Instead of "creating an empty .ser file" which is later populated with information from the HUB file we now load the data into the application straight from the HUB file itself.

When selecting a module instead of showing the module data on the overview dashboard like thought previously we now have a separate window "Module Overview" which displays the information for the selected task. We changed this due to the fact the overview dashboard would become too cluttered and ruin user experience

Milestones are now attached to Assessments and not Modules as stated before. This was changed due to the change in our understanding of the clients requirements. Also the precondition changed to only allow milestones to be created once a task(s) exists within the assessment the milestone is trying to attach to. This was because if you created a milestone without any tasks attached it would automatically be complete because milestones are completed once all related tasks are completed.

Everything to do with attaching tasks to milestones and attaching activities to tasks is now done via the corresponding add form. E.g. The activity form allows the user to select from a list of applicable tasks to attach it to. Notes however remain the same and can be attached after an existing task/activity has been created.

Viewing the gantt chart can now only be accessed via a module window and will display information for only the module selected.

Selecting a module is now via the overview dashboard in the form of a clickable list which takes you to a module window with the information or via a side nav bar which allows the user to switch between modules

A new use case "Save file" has been added which was not thought of in our initial design but turned out to be very important. This use case explains the steps a user takes to save their semester profile to disk

OMT Class Analysis

To identify classes and attributes of the system, the first stage is to perform an OMT (Object Modeling Technique) Analysis of the project specification to identify key nouns that could be classes/attributes, and then identify verbs that represent relationships between these candidate classes.

Stage 1 - Identify nouns in Scenario

- Module Organiser
- Student
- UEA

- Study Planner Application
- Semester Study Profile
- Button
- Module Schedule
- Dependencies
- Start date
- end date
- module
- Coursework assignment
- Task progress
- deadline
- Gantt Chart
- milestones
- HUB
- Data File
- Task
- Activity
- progress dashboard
- semester
- Error
- module content area
- Exams
- weightings
- Assessment
- book chapters
- assignment requirements
- notes
- Extension
- Activity Type
- Progress bar

Stage 2 - Identify candidate classes and attributes

Kept Classes

- **Organiser**
  - The module organise will need to update a deadline so they will be different from a student user.
- **Student**
  - The student is the main user of the system.
- **Profile**
  - Used to aggregate all the module schedules in a semester
- **Schedule**
  - Used to aggregate times for exams, assignments, etc. to create a timetable
- **Tasks**

- ○ An assessment can to have tasks and a task will require different attributes to properly display.
- **Module**
  - ○ Semester has modules, which will contain information on module assessments, deadlines, lecture dates, etc.
- **Milestone**
  - ○ A milestone contains its own attributes so will be its own class.
- **Gantt Chart**
  - ○ Each module will have its own gantt chart object so it will need to be in its own class.
- **Activity**
  - ○ A task can have activities which will store their own attributes.
- **Coursework**
  - ○ This will store details on a piece of coursework associated with a module
- **Exam**
  - ○ This will store details on a modules exams.
- **Assessment - Abstract**
  - ○ This will be an abstract class because exam and Coursework will have similar attributes so they can inherit from this.
- **Note**
  - ○ Both Tasks and activities have notes, which store their own attributes.

Kept Attributes

- **Dependencies**
  - ○ A task may have a dependency which will be just be a reference to another task.
- **Start date**
  - ○ Essential to many classes, e.g. task, activity, schedule, etc. but doesn't need to be its own class.
- **End date**
  - ○ Similar to start date, end date is essential to many classes, e.g. task, activity, schedule, etc. but doesn't need to be its own class.
- **Task Progress**
  - ○ stores the percentage of progression to a task. As task activities are completed the progress is updated.
- **Deadline**
  - ○ This just stores a date deadline for an assignment.
- **Activity Type**
  - ○ Stores the type of activity being added to a task. e.g. studying, programming, writing, etc.
- **Weighting**
  - ○ this stores the value of exams and coursework contribution to the overall module grade.
- **Task Requirements**
  - ○ stores the requirements for a task, e.g. time studied, book chapters covered, assignment requirements completed.

# Stage 3 - Discard spurious classes

Discarded Classes

- UEA
- Study Planner Application
- Button
- HUB
- Data File
- Progress Dashboard
- Semester
- Error
- Module content area
- Book chapters
- Extension
- Progress bar

Class Relationships

- Activities include start date, end date, module, coursework assignment, task progress, Dependencies
- User creates study profile
- User selects module
- User defines study tasks
- Study tasks belong to an assessment event
- Task depends on previous task
- Task can have notes
- User defines milestones
- Milestones related to tasks
- Module organiser updates deadline
- User defines study activities
- Activity is attached to a task
- Activities can have notes

Remaining Classes
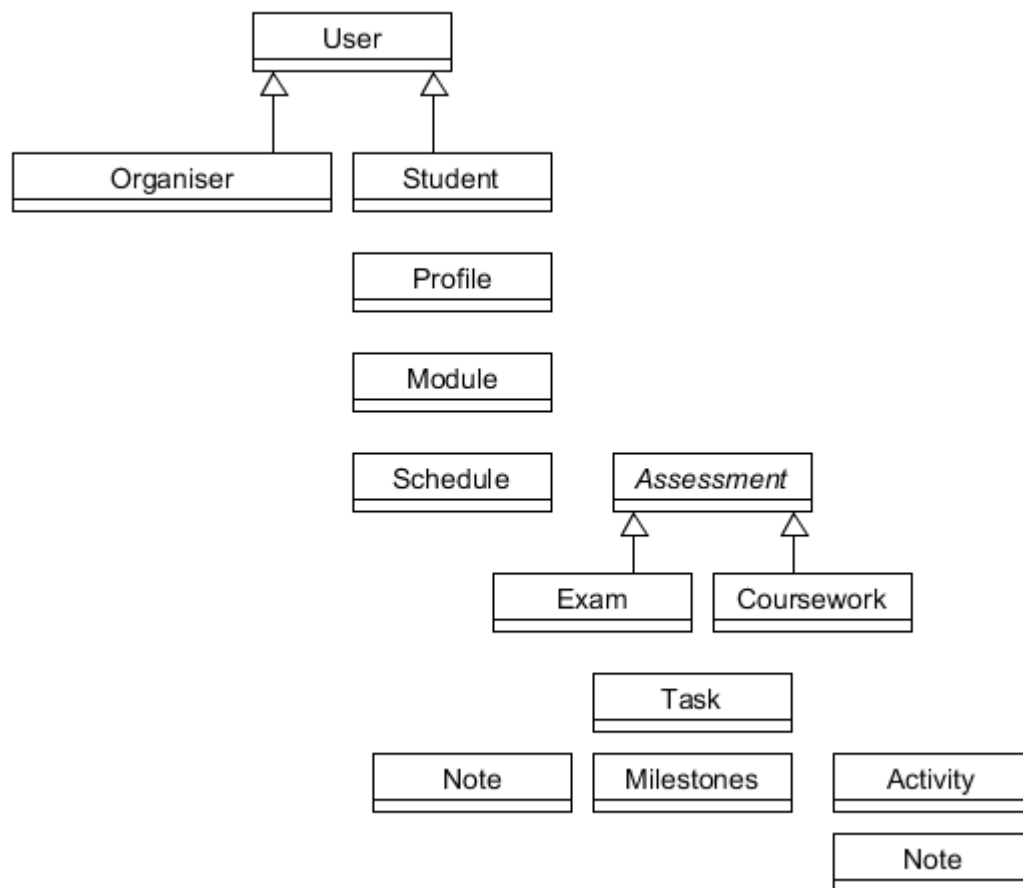
This image shows a basic representation of our approach to the problem. We have decided to use a tree like structure which means every class in the tree only has access to classes below it.

A student has a profile which contains many modules. Each module has a schedule and a set of assessments. Each assessment can have a task attached to it and each task can have an activity, note or/and a milestone attached to it. Activities can also have note attached.
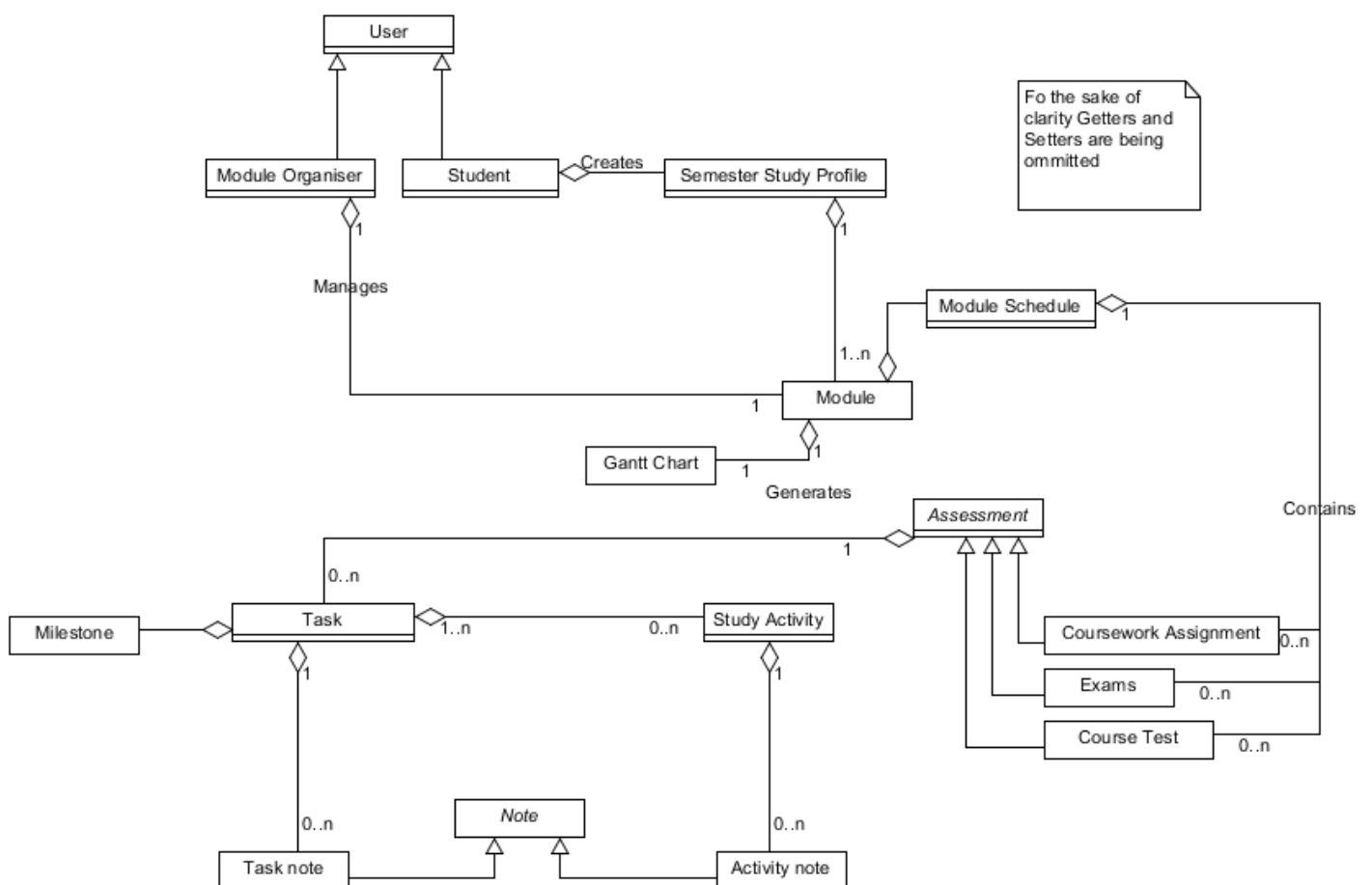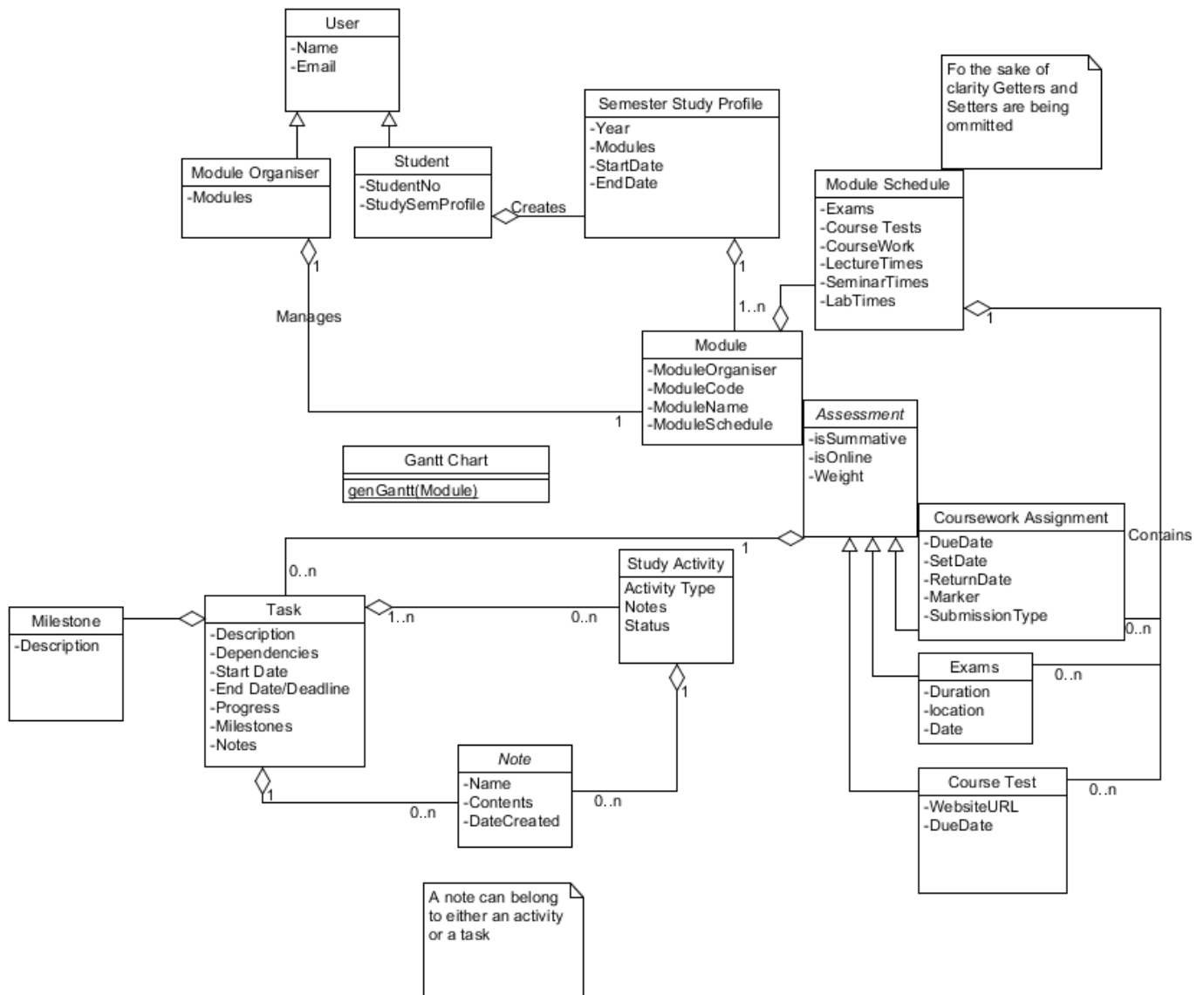
Object Oriented Analysis

Objects and Relationships

Using the results of our OMT analysis we produced a class diagram that shows the classes that are relevant to the problem domain and the relationships between the classes that we have identified.

Objects, Relationships and Fields

We then added the attributes that we identified in the OMT analysis as well as adding our own that were relevant to each class.

User
-Name
-Email

Module Organiser
-Modules

Student
-StudentNo
-StudySemProfile

Creates

Semester Study Profile
-Year
-Modules
-StartDate
-EndDate

Fo the sake of
clarity Getters and
Setters are being
ommitted

Module Schedule
-Exams
-Course Tests
-CourseWork
-LectureTimes
-SeminarTimes
-LabTimes

Manages

1..n

Module
-ModuleOrganiser
-ModuleCode
-ModuleName
-ModuleSchedule

1

Gantt Chart
genGantt(Module)

Assessment
-isSummative
-isOnline
-Weight

Coursework Assignment
-DueDate
-SetDate
-ReturnDate
-Marker
-SubmissionType

Contains

Study Activity
Activity Type
Notes
Status

0..n

1

1..n

Task
-Description
-Dependencies
-Start Date
-End Date/Deadline
-Progress
-Milestones
-Notes

0..n

Milestone
-Description

0..n

Exams
-Duration
-location
-Date

0..n

Note
-Name
-Contents
-DateCreated

1

0..n

0..n

Course Test
-WebsiteURL
-DueDate

0..n

A note can belong
to either an activity
or a task

29

Object Oriented Design

Since our previous design class diagram was so unlike our final code base we reverse engineered our code using EasyUML which is a netbeans plugin. The resulting image was too large to put in this report while still being legible so it has been included in the doc folder of the netbeans project where this report is also located.

System Architecture

Once we started using the Swing form builder we realised that Swing is both the view and Controller. It uses its component hierarchy like JFrame and JList to provide the view section of the MVC architecture and the builder also hides a lot of the controller code which uses Java awt events in the auto generated code. The Swing builder presents us with a single method generated inside the component class for any input done on that particular component.
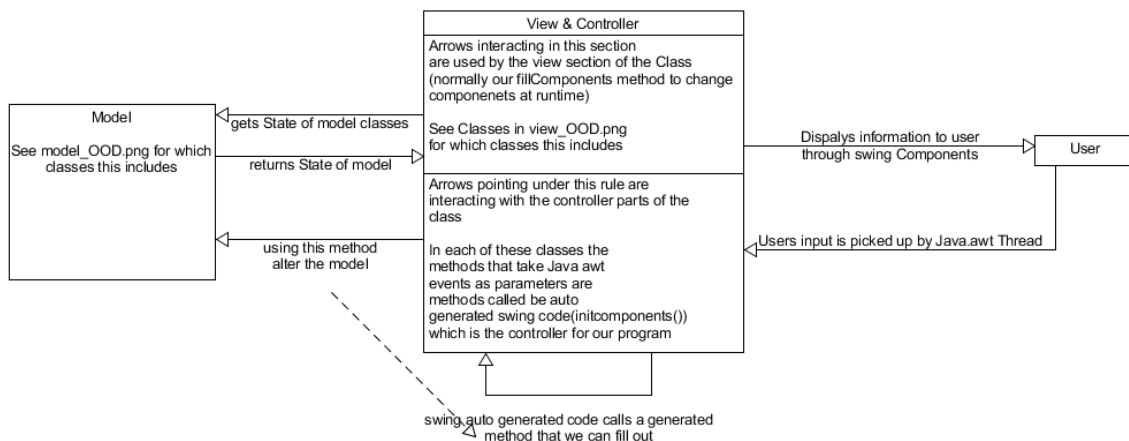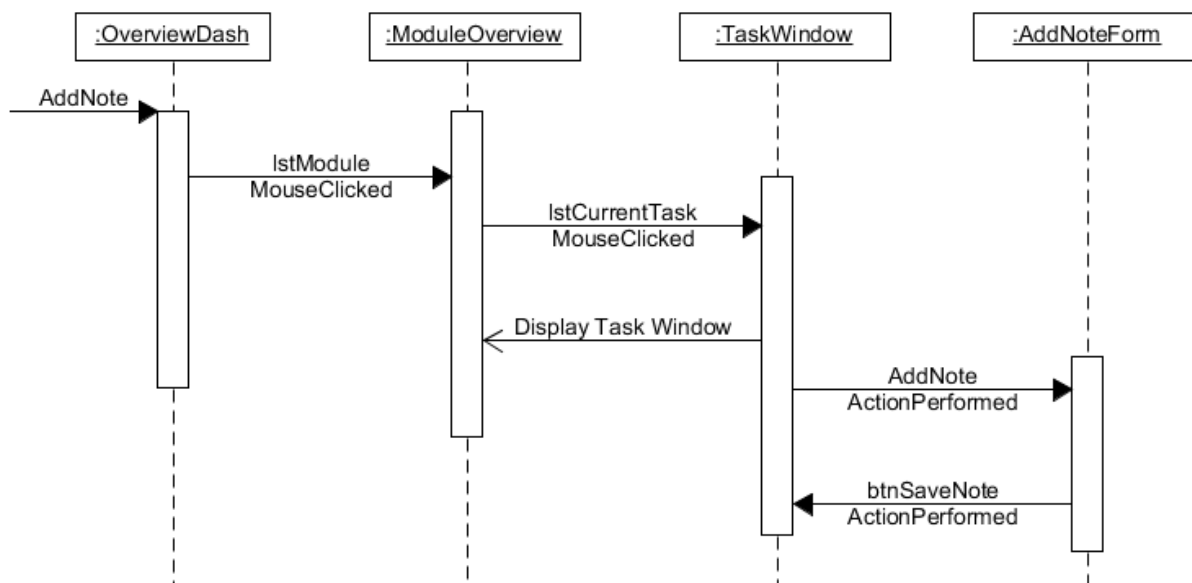
Sequence diagrams

These diagrams show the expected behaviour of 2 actions the user can perform.
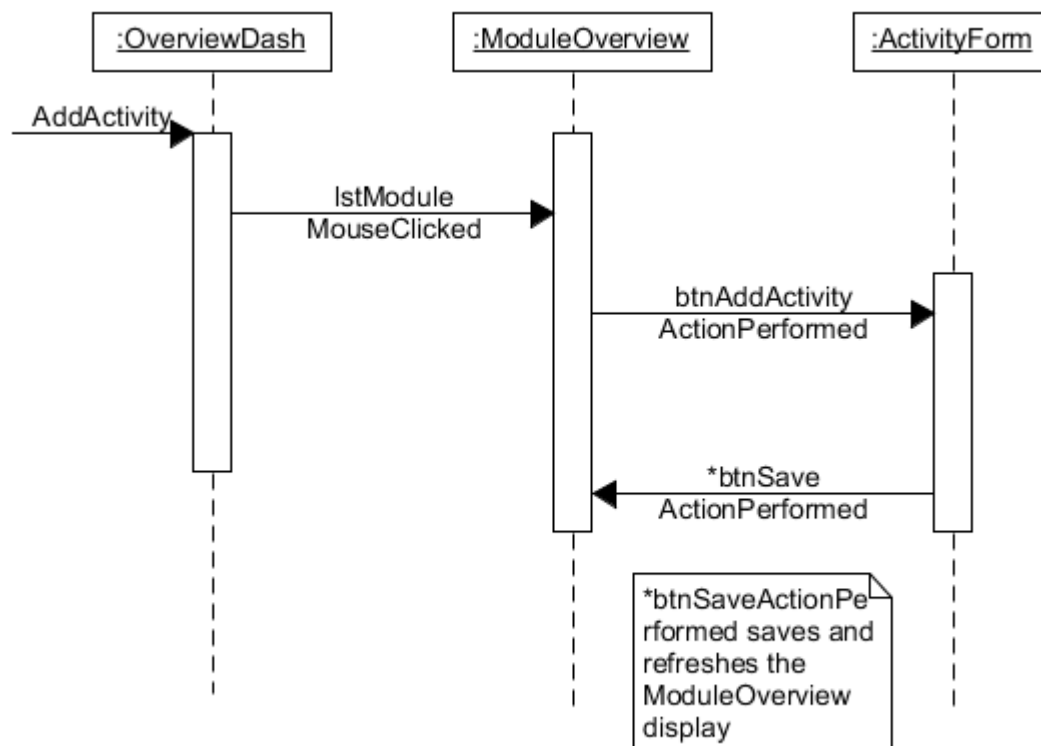
- Add Activity

The names on the Add Activity diagram have also been changed to match our function names. In the previous version the user had the Module Window displayed in the Overview Dashboard, however we changed that so the user instead can select a module from the Overview Dashboard, which opens a new window called Module Overview which displays the information.


- Add Note

In the Add Note sequence diagram we changed the way that users access the note form. In the previous version the user had to go through the Module Page -> Task -> Activity -> Note Form which resulted in the user having to navigate through a lot of windows just to add a note. In the final version the user is able to add notes straight from the Task Window. We also changed the naming on the diagram to match our function names.

State Diagrams

Window state diagram final
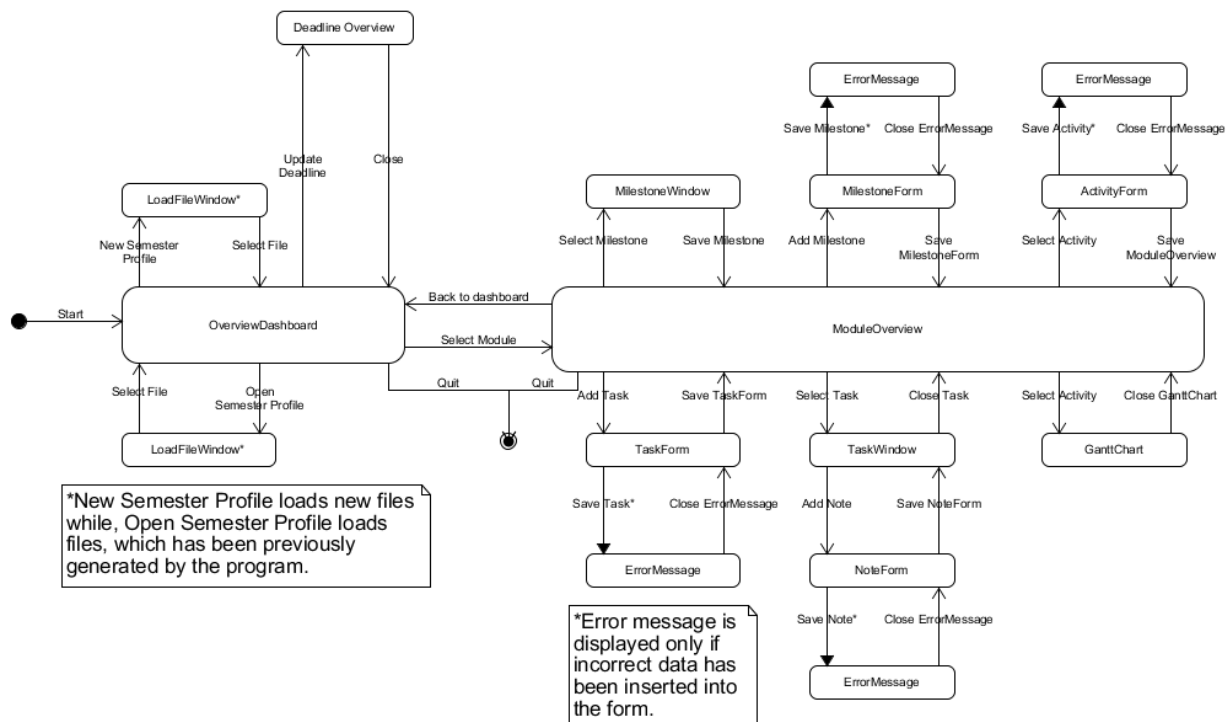
Something that may not be obvious when looking at the diagram is that the Module Overview window has a nav bar on the side which allows the user to switch between the modules and also provides the link to go back to the dashboard

We updated this from our original design in a number of ways. Firstly we made sure to add in all the forms which included Task form, Activity form and Milestone form. These are all accessed via clicking the corresponding buttons the Module Overview window. All of these forms will cause an error window to pop up when information is either missing / incorrect. Once the error window is closed the task form will be displayed with the information previously entered as well as highlighted areas that caused the error. A note form was also added which is accessed via the Task Window which behaves in the same way.

The ability to access a gantt chart form the overview dashboard was removed because it was too complex to have a gantt chart that visualizes everything and was decided that having a gantt chart for each module which could be accessed via the module overview window was a better overall idea for our system. A note form was also added which is accessed via the Task Window which behaves in the same way.

Functionality state diagram final
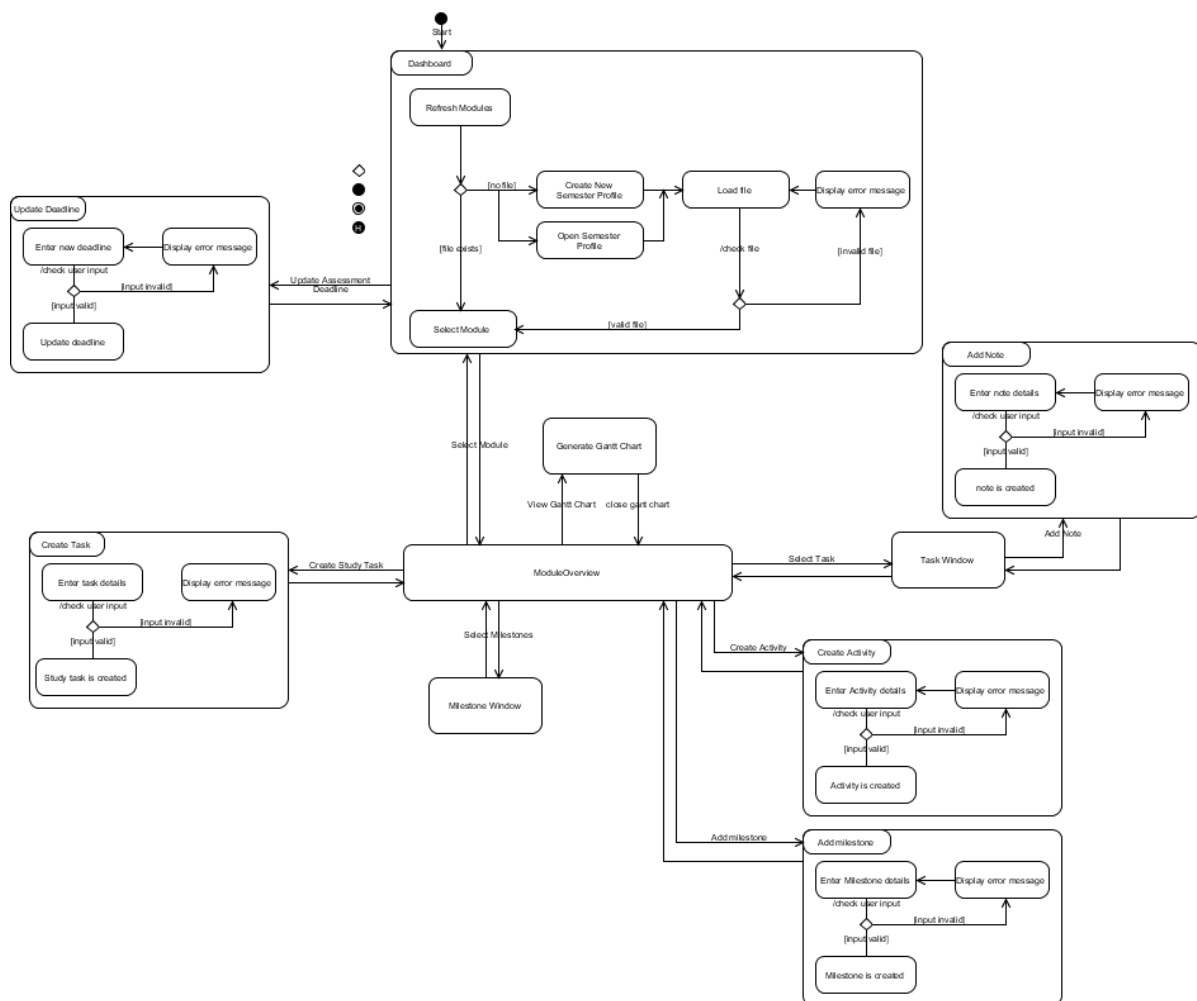
Our chosen file format

First thoughts

Our first thoughts was to use a simple csv file to store the semester profiles however this turned out to be much too simple for what we wanted to do. We had a look at using Yaml via a Java framework known as Jaml however after testing it with our system it was clear that it would not allow us to store Assessments because it did not deal well with ArrayLists and from our research did not work at all for the inheritance that was used in our design. XML was also an option however we felt that it was too complex for what we wanted to achieve and would only serve to add unnecessary bulk to our application.

Final decision

As we were given the freedom to use any file format we liked (as long as it was agreed by the user) in the end we decided to make our own file format which was similar to csv but with

added "start and end tags" which aided reading in the file. The file begins with the year and date information for that semester and then has a list of Modules which are in blocks starting with MODULE and ENDMODULE and within each module there is a list of assessments which have a start tag indicating the type e.g EXAMS. Every assessment must have a start tag however looking back if we were to do it again it would of made sense to add an end tag so assessments of the same type could be grouped. This was not thought of during the design due to the fact that we knew that each module was not going to have more than 4-5 assessments.

Saving semester profiles
When exporting a semester profile with all of the users tasks, activities and milestones we used the Serializable interface in java in-order to serialize the model and make it possible to load it back in via a button on the overview dashboard. The user has the option to name their save files which allows multiple users to use the application simply by loading in their own version of the semester profile.

If the user loads in a New semester profile then once they exit the application they will be prompted to give a name and save their profile. If nothing is input then the file will be saved with the default name "newSemesterProfile.ser". If the user instead opens an existing semester profile then once they exit the application the program will save using the same name as the file they opened and no file explorer will be displayed.

We decided to use the save when exit because we thought it would be a nice feature for people who tend to forget to save and therefore improve the overall user experience.

Example of our chosen file format:
2016, 01-01-2016, 31-05-2016
MODULE
Mr Guy, CMP-5014A, Programming
COURSEWORK
Java 1, 10-04-2016, 27-03-2016, 15-04-2016, Mr Mark, hand into hub, true, false, 25
COURSEWORK
C++ 1, 28-04-2016, 16-04-2016, 29-04-2016, Mr Fail, upload to blackboard, true, false, 25
EXAMS
Programming Exam, 120, CHALL, 04-05-2016, true, 50
ENDMODULE

Group Working Process

During the implementation of our system we found that the best way to work was for the four of us to sit around one or two computers and pair-programming together. This way we could discuss everything as it was implemented and were able to catch any issues or errors much faster than if we worked individually and only discussed after we had implemented large chunks of the program. Working this way meant that we communicated better, everyone being able to voice their opinions and concerns and we were able to discuss the best course of action there and then. This also means that on our BitBucket Repository most commits will be performed by the same person, but in the comments we have specified exactly who contributed to that commit.

Additionally, this method of pair-programming means that we can share our knowledge base, so that if one team member knows more on Java Swing than the others, instead of them being given the majority of that work to do, we work together so that everyone learns from them and is able to contribute themselves.