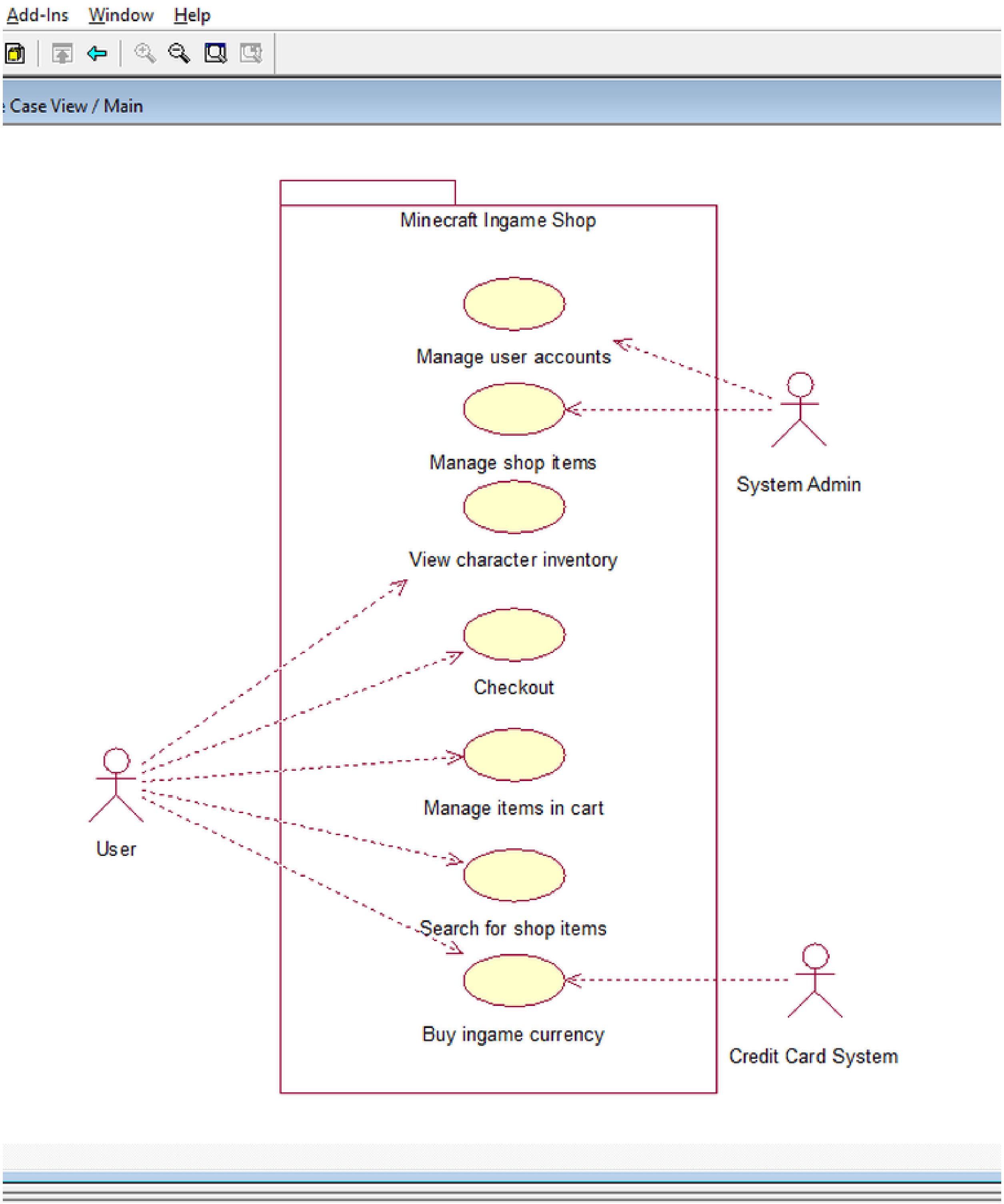
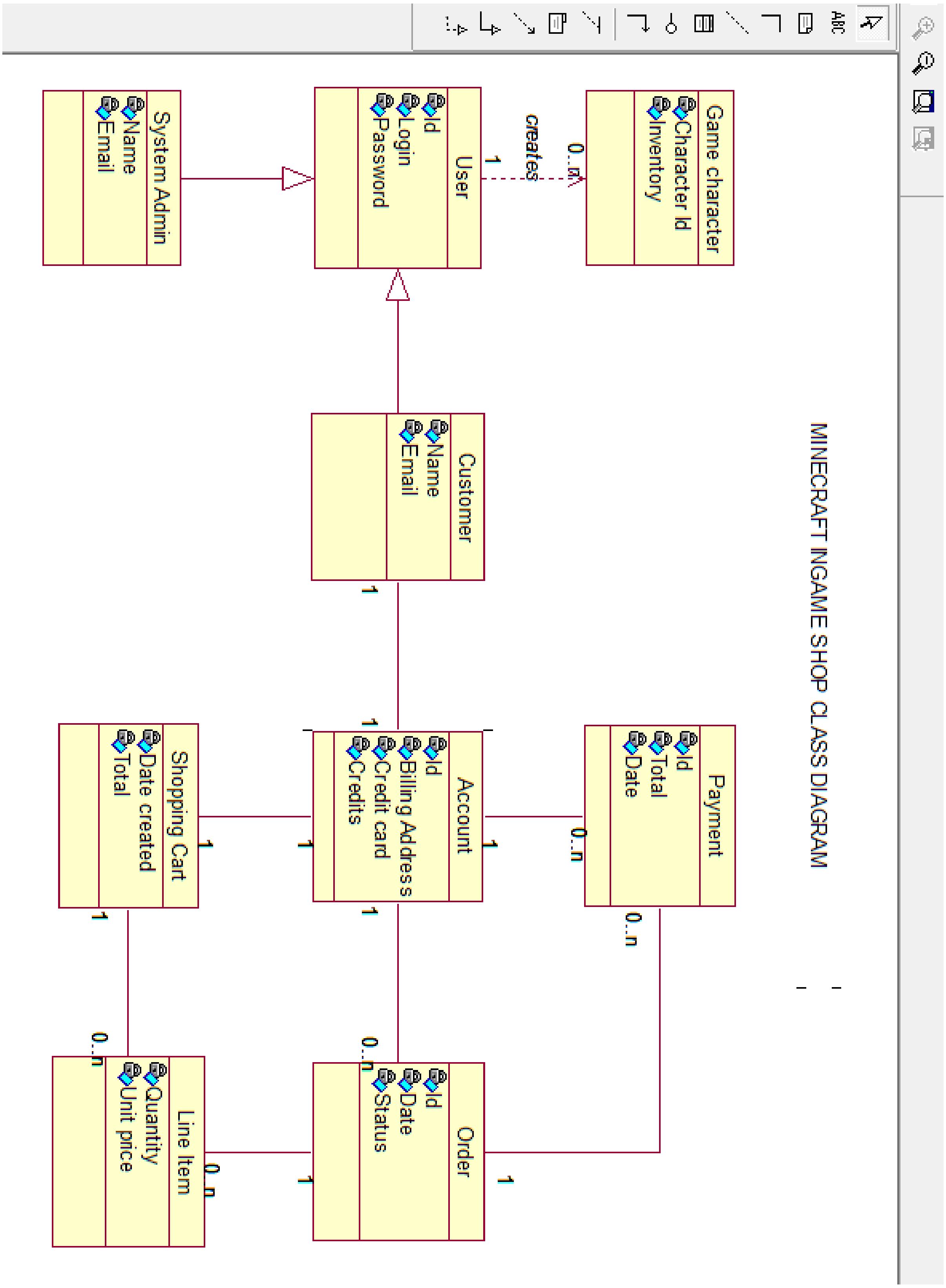
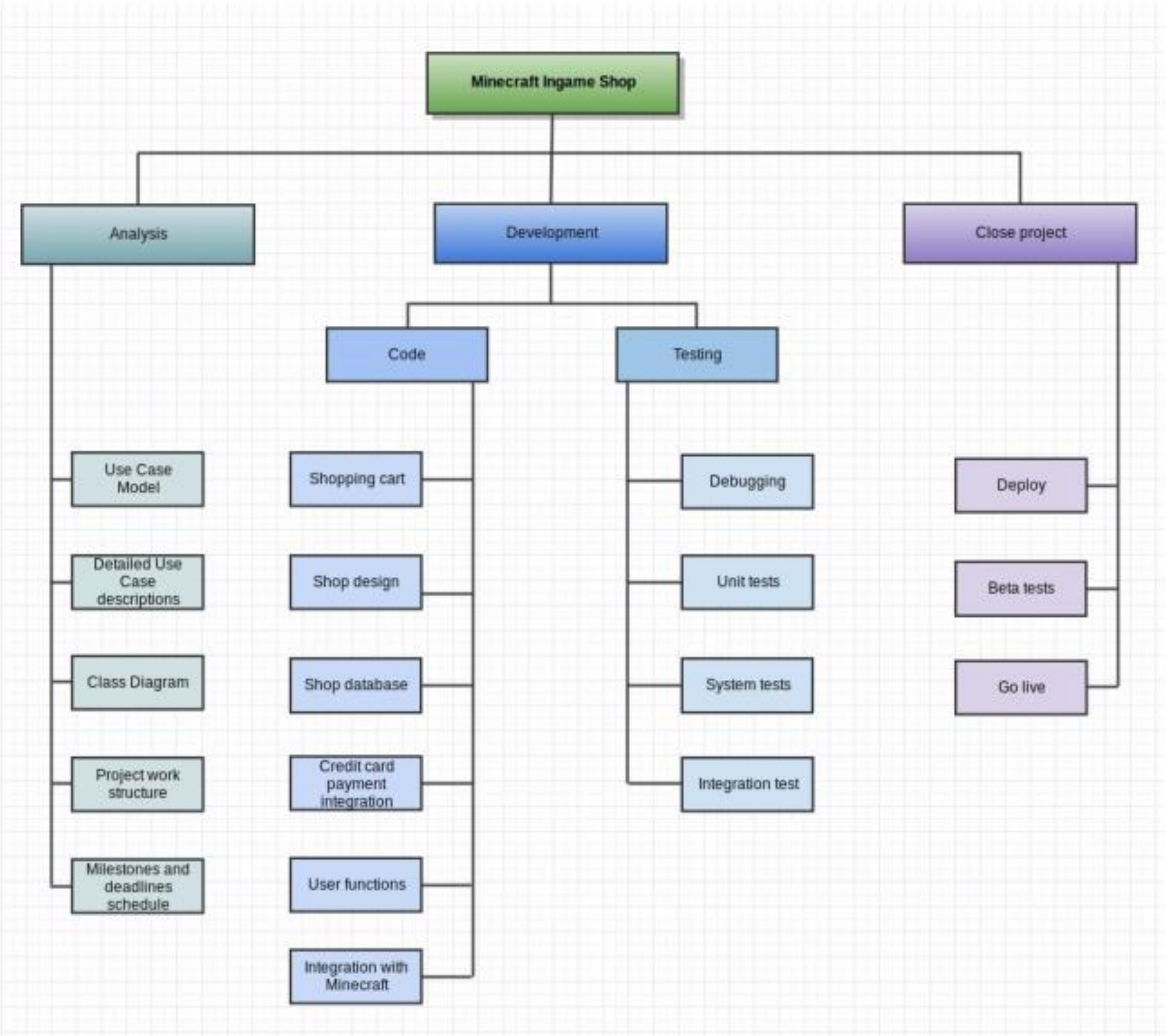
ii) Use Case Model

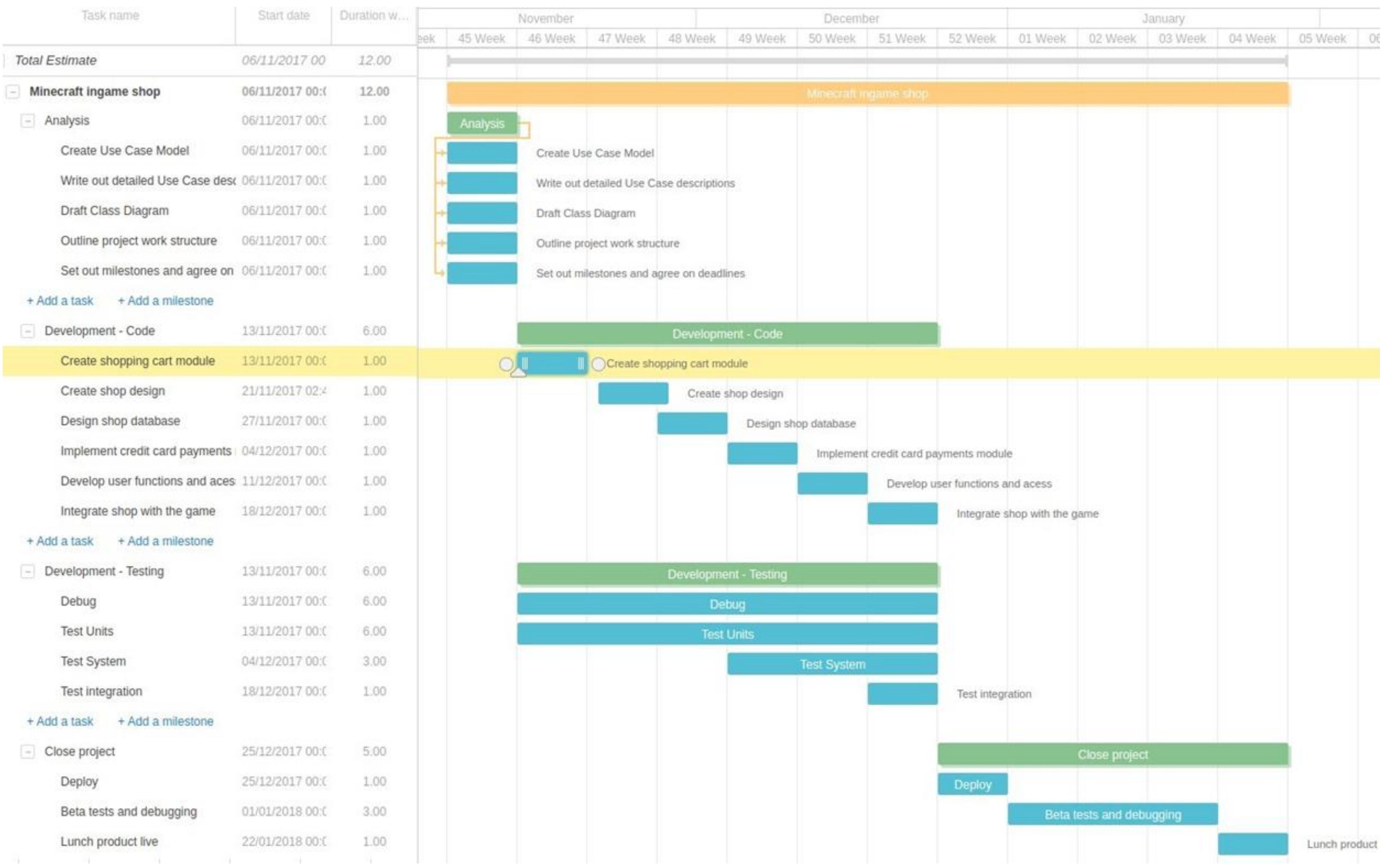


iv) Conceptual class diagram



v) Project Plan





Task name	Start date	Duration w...
Total Estimate	*06/11/2017 00*	*12.00*
⊟ **Minecraft ingame shop**	**06/11/2017 00:(**	**12.00**
⊟ Analysis	06/11/2017 00:(	1.00
Create Use Case Model	06/11/2017 00:(	1.00
Write out detailed Use Case des(	06/11/2017 00:(	1.00
Draft Class Diagram	06/11/2017 00:(	1.00
Outline project work structure	06/11/2017 00:(	1.00
Set out milestones and agree on	06/11/2017 00:(	1.00
+ Add a task + Add a milestone		
⊟ Development - Code	13/11/2017 00:(	6.00
Create shopping cart module	13/11/2017 00:(	1.00
Create shop design	21/11/2017 02:4	1.00
Design shop database	27/11/2017 00:(	1.00
Implement credit card payments	04/12/2017 00:(	1.00
Develop user functions and aces	11/12/2017 00:(	1.00
Integrate shop with the game	18/12/2017 00:(	1.00
+ Add a task + Add a milestone		
⊟ Development - Testing	13/11/2017 00:(	6.00
Debug	13/11/2017 00:(	6.00
Test Units	13/11/2017 00:(	6.00
Test System	04/12/2017 00:(	3.00
Test integration	18/12/2017 00:(	1.00
+ Add a task + Add a milestone		
⊟ Close project	25/12/2017 00:(	5.00
Deploy	25/12/2017 00:(	1.00
Beta tests and debugging	01/01/2018 00:(	3.00
Lunch product live	22/01/2018 00:(	1.00

v. Monitor project

I would start by designing monitoring process into the project itself.

Monitoring would start right at the beginning of the projects, and continued throughout its development life.

Monitoring plan would include documentation that specifies what elements (and how) should be measured (the critical one's). Further, documentation would outline what performance level would be satisfactory, frequency of the reporting, and communication procedures.

I would have designated people within the team, whos responsibility involves monitoring progress, and communicating issues. Further, I would ensure that there's a clear structure outlining who reports to whom.

As the project progresses, deliverables would be measured against set goals, and corrective action would be taken when necessary. For example, if deliverable isn't performing as expected, the plan would be revised to reflect necessary changes. This way, project would stay on track, and account for unexpected changes.

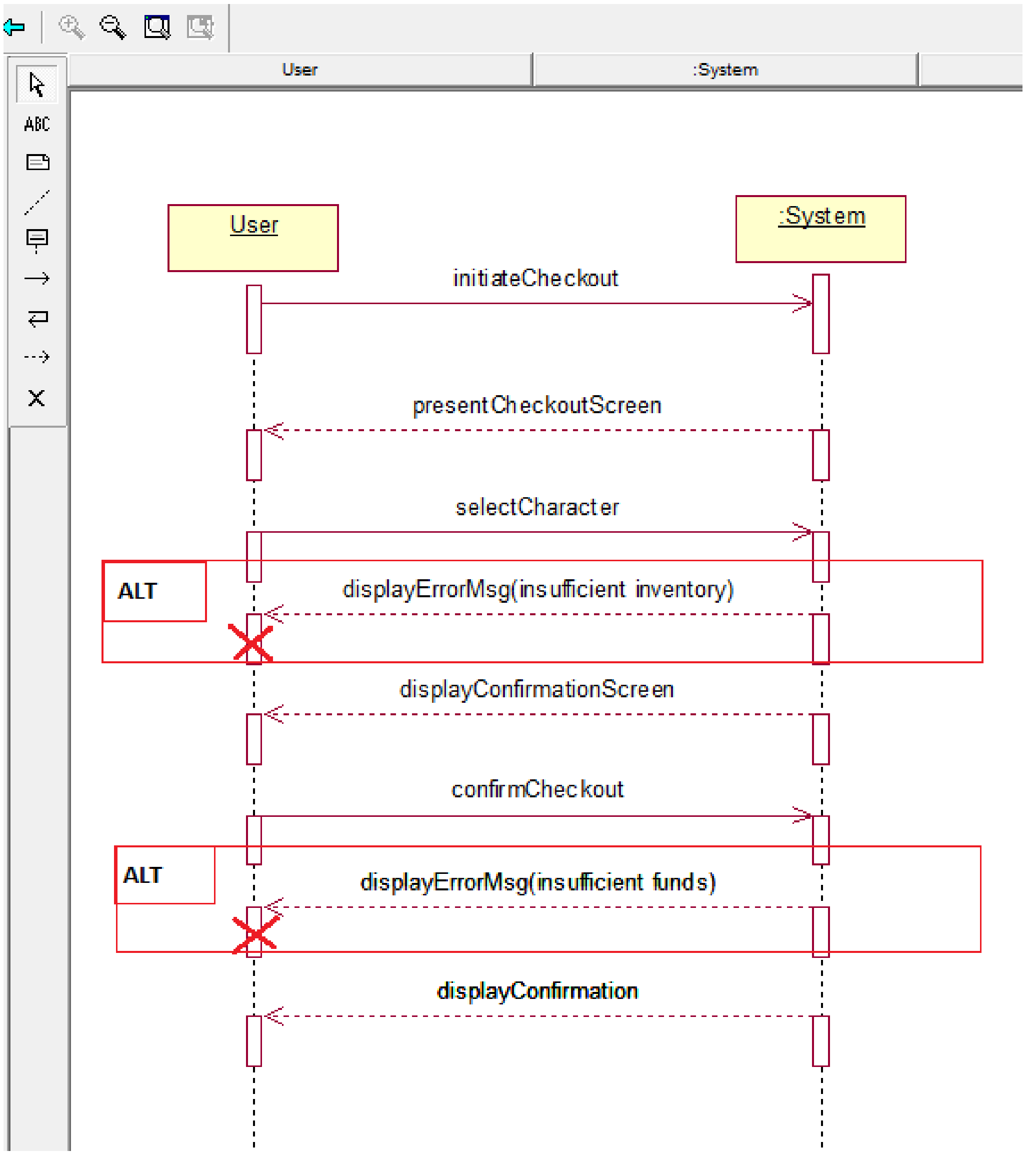
Finally, there would be person responsible for ensuring that project doesn't goes beyond planned spectrum. For example, making sure that additional features aren't introduced into the project, unless they are absolutely needed. This would assure that project doesn't spiral out of original scope.

vi) Glossary

Term	Category	Comments
Payment	Use case	Process that allows user to pay for items listed in the shopping cart. Payment is done using credit card or via game credits.
Manage items in cart	Use case	Process that allows user to add / delete items in shopping cart.
Account	Class	Class that holds customers credit card details, purchased items , and payments made.
User	Class	Abstract class, that unifies common attributes. Refers to any person using the system (customers and system admins).
Line item	Class	A ‘virtual’ item that can be bought in the store. It is Minecraft related, and can be used by user’s characters in game.
Password	Attribute	Password used by user to log in to the shop.
Inventory	Attribute	List of all items owned by the game character including number of free slots available.
Credits	Attribute	Number of credits available to account user. Credits are ingame currency, and can be bought using Payment system.
Update info	Method	Method for allowing user to update his contact details / email address.

vii) System Sequence diagram

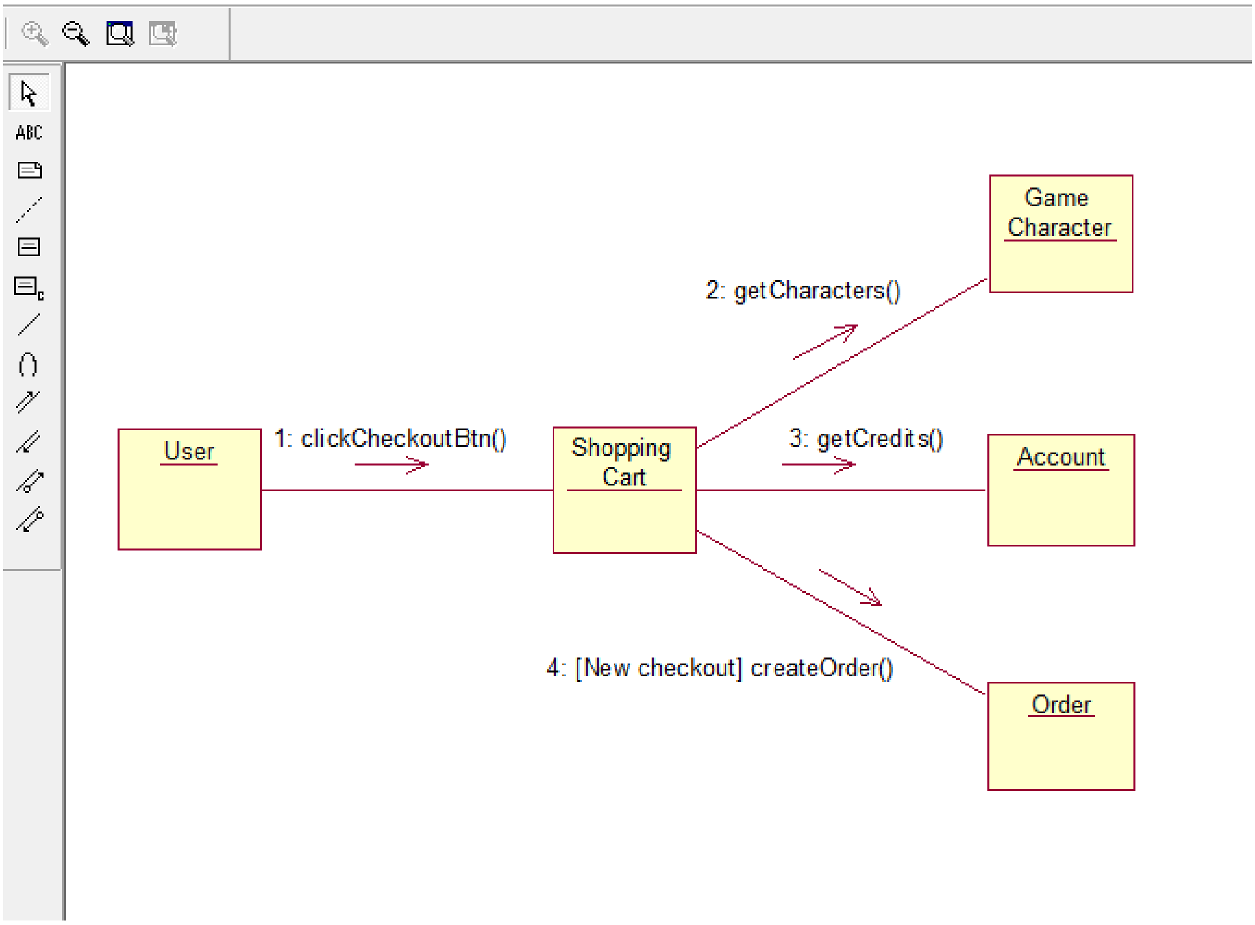
Based on use case 'Checkout'



viii) Contracts & ix) Communication diagrams

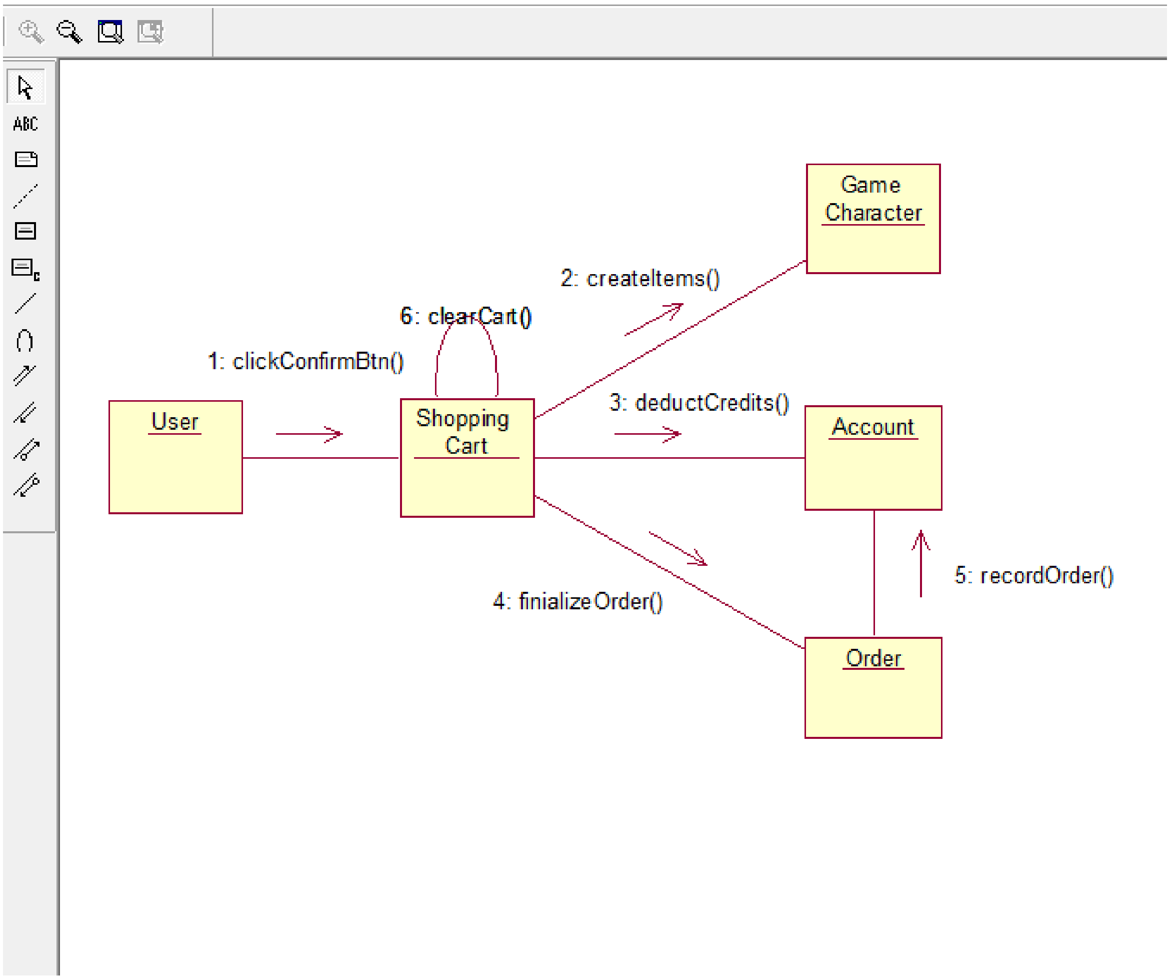
a) **initiateCheckout()**

Name	initiateCheckout()
Responsibilities	When user clicks a 'Checkout' button, start a new sub window with item's totals listed.
Type	System
Pre-conditions	There is at least one item in the shopping cart.
Post-conditions	If no current checkout active, new order initiated.



b) **confirmCheckout()**

Name	confirmCheckout()
Responsibilities	Once user confirms purchase, finalize order, display summary message, create new items in selected character's inventory, deduct credits from user's account, and finally, clear shopping cart.
Type	System
Pre-conditions	Checkout initiated. New order exists. Selected characters have enough inventory space.
Post-conditions	Deduct credits from user account. Order recorded in user Account, status set to 'finalized'. New items created, and added to inventory. Shopping cart cleared.



xii) Testing plan

Testing will be an important part of the project itself, and will be used to verify that the deliverables meet project plan specifications. Project manager will ensure that test plan has been developed at the beginning of the project. Test manager will be responsible for overseeing testing throughout the project.

As the project progresses, each deliverable will be subject to unit testing to ensure that the code does what intended. Unit tests look at small portion of code in isolation, and ensure inputs / outputs are as expected.

Once more units are delivered, integration testing will commence alongside unit testing. Integration tests will combine units together, and ensure that they interact with each other as intended, and that there are no unexpected results. Integration testing will become progressively bigger, and more complex as more units are being introduced.

Simultaneously to unit / integration testing, system testing will be carried out to ensure that units meet intended system requirements.

Once the applications core elements are finished, beta version of the product can be deployed and beta tests will begin. They will involve testers (users) using live product. This is a very important part of the testing, as it often highlights problems that automatic tests cannot detect.