University of Glasgow | Department of Computing Science

D6: SYSTEM DESIGN AND ACCEPTANCE TEST PLAN

Deliverable ID	D6
Deliverable Title	System Design and Acceptance Test Plan
Project	PSD3 Group Exercise 2
Team	V
Authors	Ross Adam Andrew Gardner Nicole Kearns Mamas Nicolaou Asset Sarsengaliyev
Deliverable Date	30th January 2013
File Name	D6.tex
Version	0.7

Contents

1 Introduction

1.1 Identification

System Design and Acceptance test plan for internship management system.

1.2 Related Documentation

PSD3 Group Exercise Description `http://fims.moodle.gla.ac.uk/file.php/128/coursework/psd3-ge-1-rev3278.pdf`

Deliverables Template `http://fims.moodle.gla.ac.uk/file.php/128/coursework/templates.zip`

PSD3 Course Notes `http://fims.moodle.gla.ac.uk/file.php/128/lecture-notes/notes-r3275.pdf`

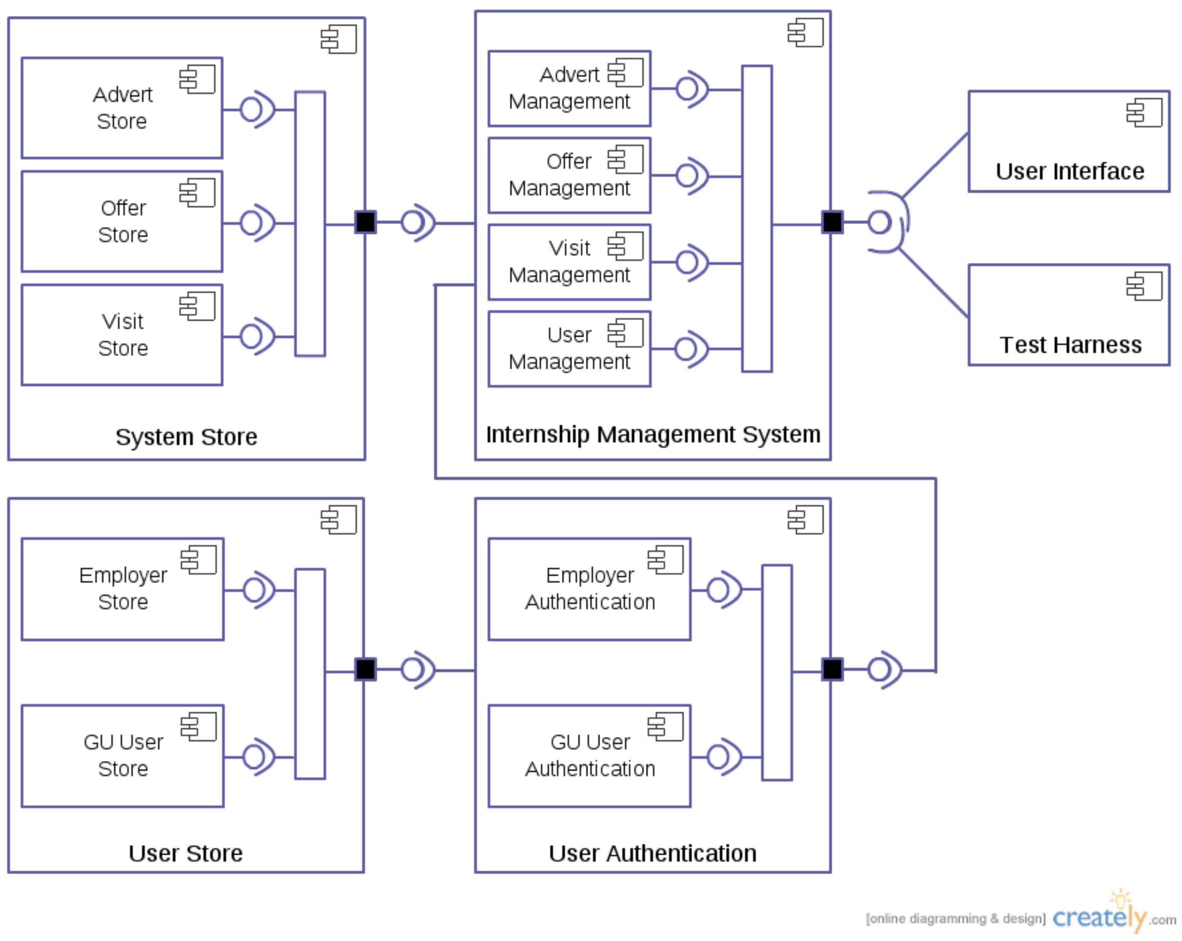
1.3 Purpose and Description of Document

The purpose of this document is to give a detailed description of our system desgin and acceptance plan for the internship management system. This report includes a componenet diagram showing the overall architecture of the system, state charts showing the transition between states for the different entities maintained in the system, a set of class diagrams (one for each component in the design), an API specification and an acceptance plan describing the test cases for each component in our design.

1.4 Document Status and Schedule

Date	Change	Version	Author
29/01/13	Added the state diagrams	0.1	Ross
30/01/13	Added the rationale and description for state diagrams	0.2	Ross
30/01/13	Added the information to the introduction section	0.3	Nicole
30/01/13	Added the APIs for the Stores and Authentication	0.3	Nicole
30/01/13	Added the component diagram, and the description and rationale	0.3	Nicole, Ross
30/01/13	Added the Acceptance tests	0.4	Asset
30/01/13	Updated the Acceptance tests	0.5	Ross
30/01/13	Added the APIs for the Advert, Visit, Offer and User Management	0.6	Andrew
30/01/13	Added the Class Diagrams	0.7	Mamas
12/02/13	Fixed minor errors - like spelling mistakes	0.8	Nicole
19/02/13	Updated the acceptance test plan	0.9	Nicole
19/02/13	Updated the API specification	0.10	Nicole

2 Component Diagram



Rationale
The component diagram was created to illustrate the structural relationship between all components of the internship management system. It also shows the modularity of the different parts of the system.

Description

The System store is comprised of three separated databases: the Advert Store, Offer Store and the visit store and each of these are drawn together using a facade, which will direct queries to the relevant database.

The User Store contains employer store and GU user store. The employer store is a custom database and the GU user store is a front-end to the myCampus system. Interaction with the myCampus system will be limited to retrieving basic student and staff information.

The User Authentication uses two separate components to forward the data to the relevant login system: myCampus or the employer store.

The Internship Management System contains several different components that allow the user to interact with the various other components and data stores within the system.

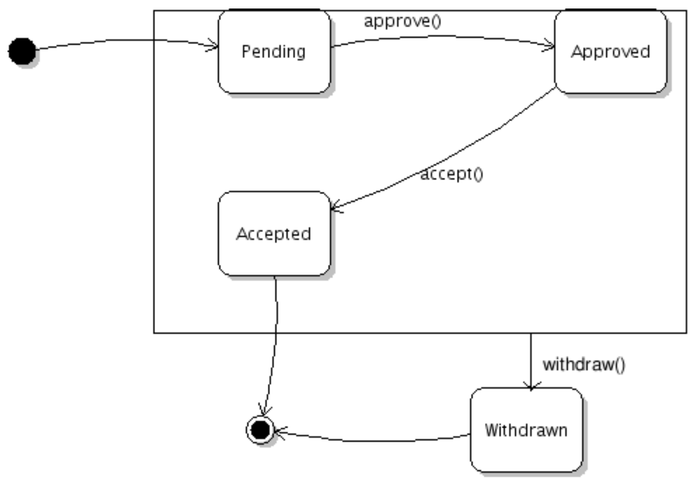
3 State Diagrams

3.1 Rationale

The following three diagrams were created to illustrate how entities within the application change over time. The three entities are Student, Internship and Advert. These entities were identified by the "status" variable in their specification.

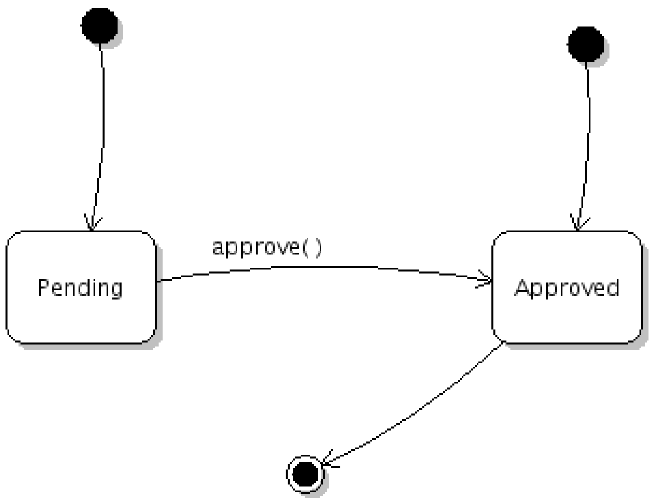
3.2 Student

Description
Student was choosen because it displayed 4 distinct states: Approved, Accepted, Withdrawn and Pending. These were identified from the "View Student Details" use case. However the finding other use cases to justify this proved difficult as none could be found that implied a change in the Student entity state. Therefore the following diagram was implemented in a manner deemed logical by the team. This is mostly a sub-state diagram encased within a "Not Withdrawn" box. If withdrawn then the sub-state is immediately exited.



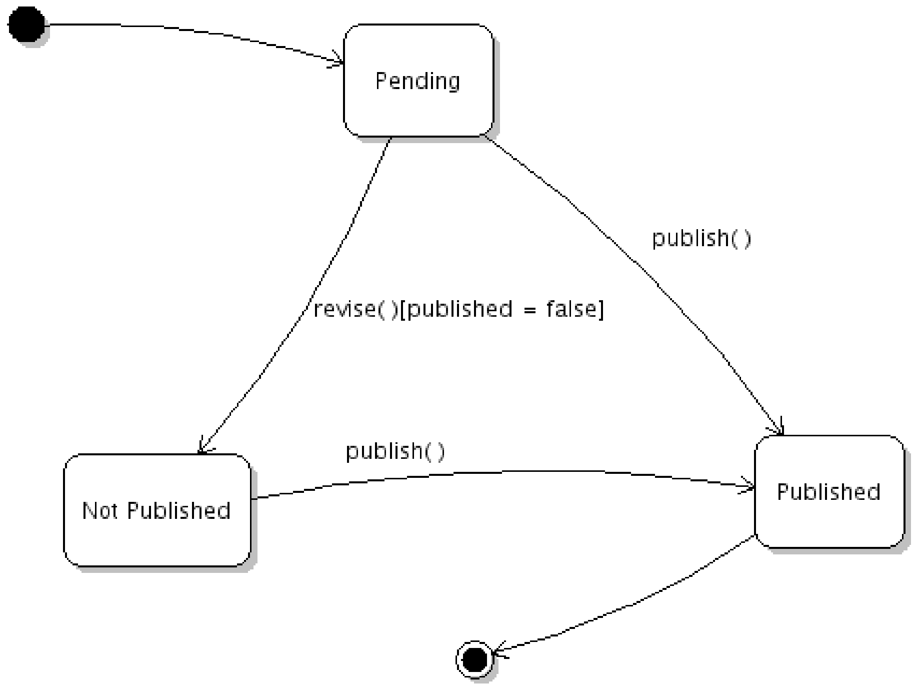
3.3 Internship

Description
The Internship has two states Pending and Approved. An Internship starts as either depending on whether it was created by the Course Co-ordinator (Approved) or by a student (Pending). A pending entity can be Approved by the approve method.



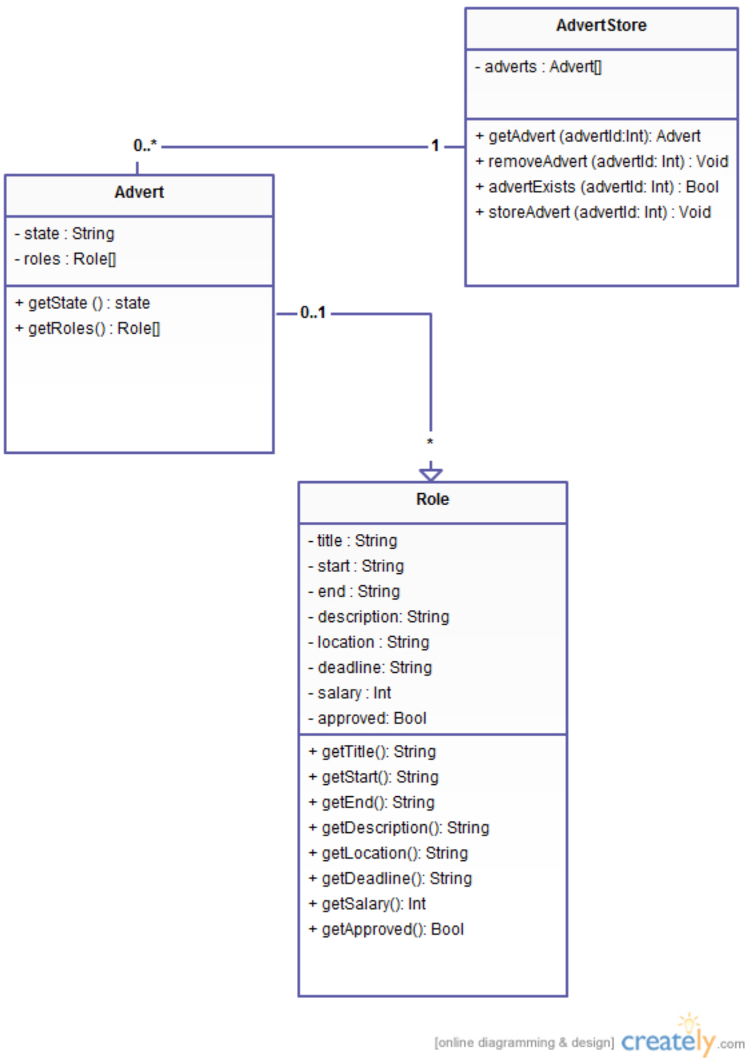
3.4 Advert

Description
An Advert can have the following states: Pendign, Not Published and Published. An advert always starts in the pending state, from there it can either become Not Published or Published. It becomes Published if it is deemed acceptable by the Course Co-ordinator and it becomes Not Published if it needs revised. Only a pending or Not Published article can be revised.

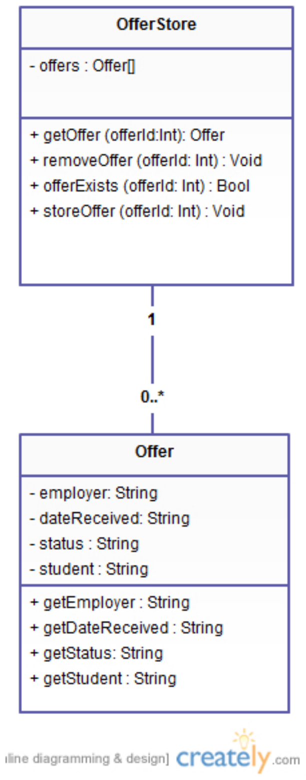


4 Class Diagrams

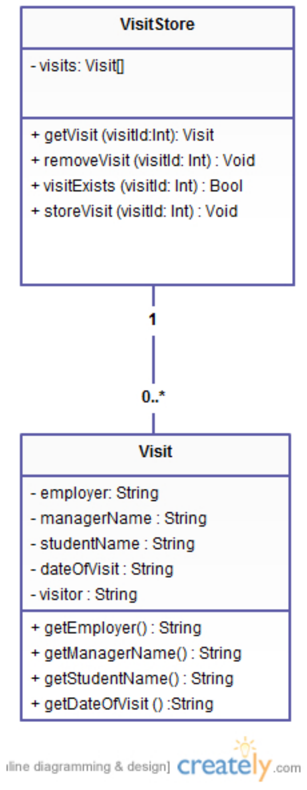
4.1 Advert Store



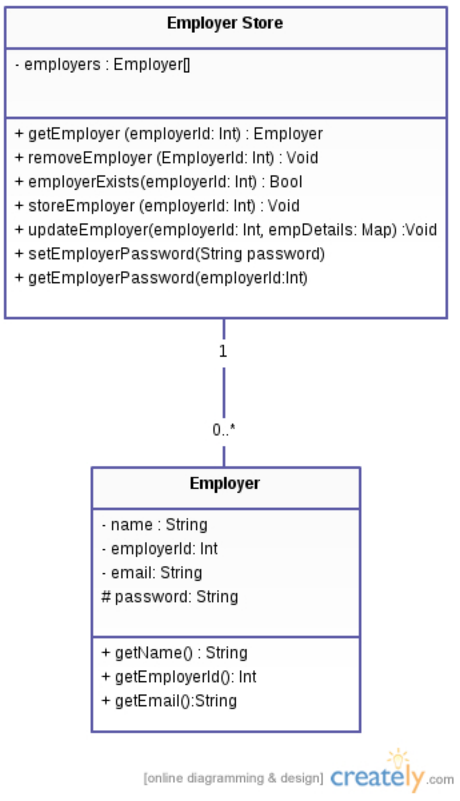
4.2 Offer Store


OfferStore
- offers : Offer[]
+ getOffer (offerId:Int): Offer
+ removeOffer (offerId: Int) : Void
+ offerExists (offerId: Int) : Bool
+ storeOffer (offerId: Int) : Void
1
0..*
Offer
- employer: String
- dateReceived: String
- status : String
- student : String
+ getEmployer : String
+ getDateReceived : String
+ getStatus: String
+ getStudent : String
line diagramming & design] creately.com

4.3 Visit Store

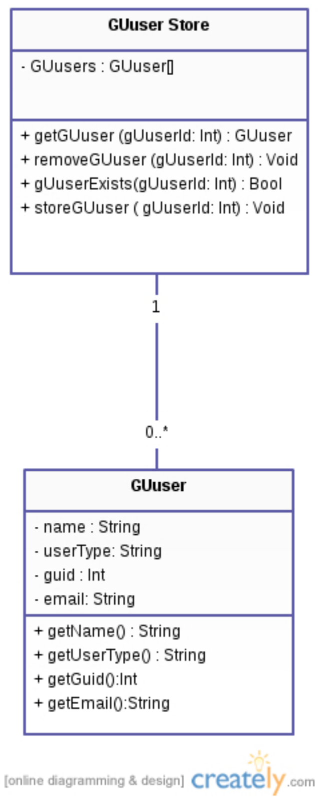

VisitStore
- visits: Visit[]
+ getVisit (visitId:Int): Visit
+ removeVisit (visitId: Int) : Void
+ visitExists (visitId: Int) : Bool
+ storeVisit (visitId: Int) : Void
1
0..*
Visit
- employer: String
- managerName : String
- studentName : String
- dateOfVisit : String
- visitor : String
+ getEmployer() : String
+ getManagerName() : String
+ getStudentName() : String
+ getDateOfVisit () :String
line diagramming & design] creately.com

4.4 Employer Store

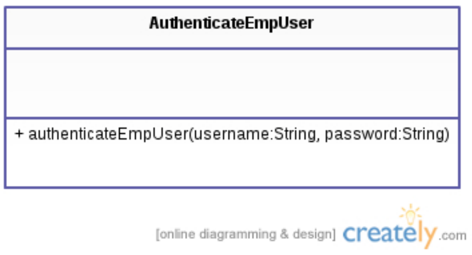


Employer Store
- employers : Employer[]
+ getEmployer (employerId: Int) : Employer
+ removeEmployer (EmployerId: Int) : Void
+ employerExists(employerId: Int) : Bool
+ storeEmployer (employerId: Int) : Void
+ updateEmployer(employerId: Int, empDetails: Map) :Void
+ setEmployerPassword(String password)
+ getEmployerPassword(employerId:Int)
1
0..*
Employer
- name : String
- employerId: Int
- email: String
# password: String
+ getName() : String
+ getEmployerId(): Int
+ getEmail():String
[online diagramming & design] creately.com

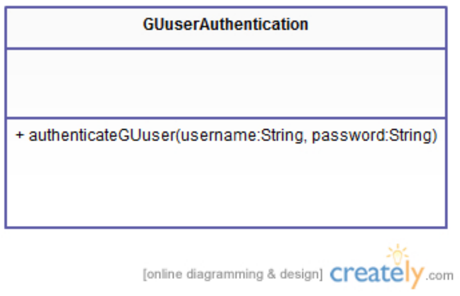
4.5 GU User Store


GUuser Store
- GUusers : GUuser[]
+ getGUuser (gUuserId: Int) : GUuser
+ removeGUuser (gUuserId: Int) : Void
+ gUuserExists(gUuserId: Int) : Bool
+ storeGUuser (gUuserId: Int) : Void
1
0..*
GUuser
- name : String
- userType: String
- guid : Int
- email: String
+ getName() : String
+ getUserType() : String
+ getGuid():Int
+ getEmail():String
[online diagramming & design] creately.com

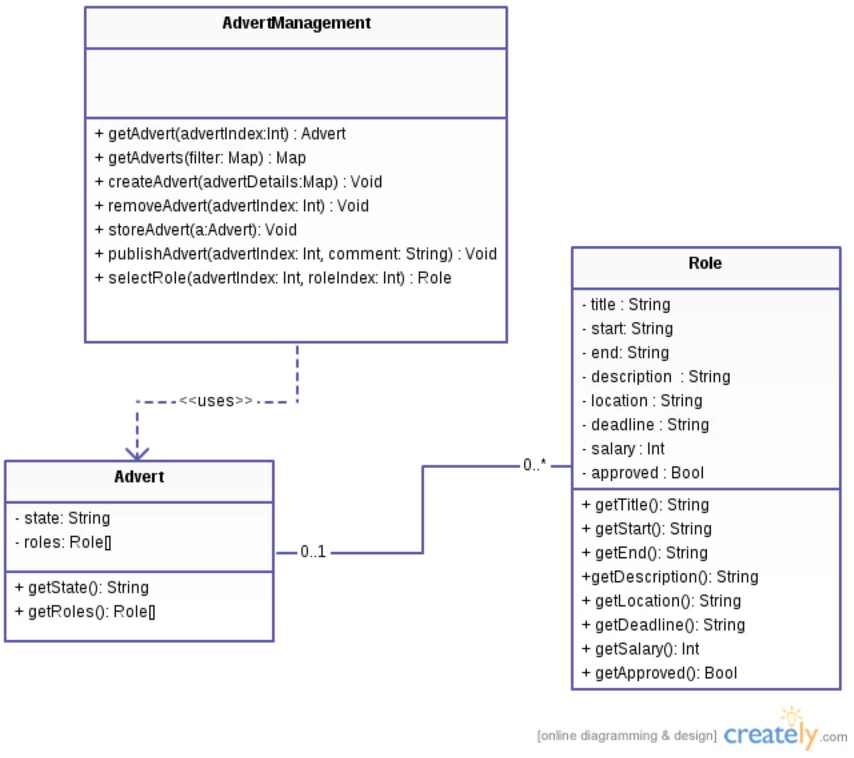
4.6 Employer Authentication


AuthenticateEmpUser
+ authenticateEmpUser(username:String, password:String)
[online diagramming & design] creately.com

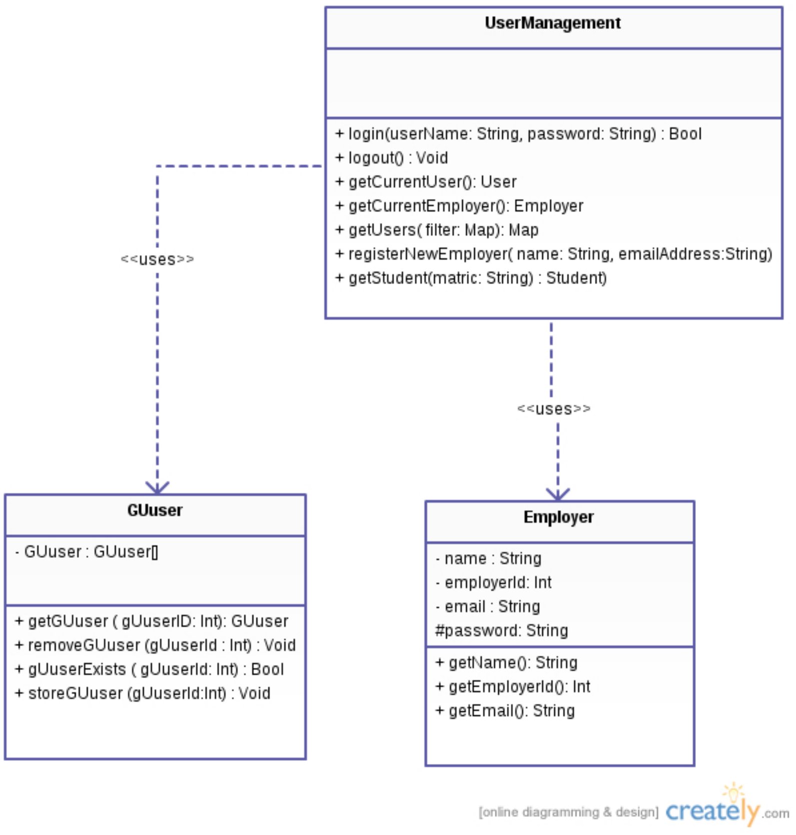
4.7 GU User Authentication


GUuserAuthentication
+ authenticateGUuser(username:String, password:String)
[online diagramming & design] creately.com

4.8 Advert Management

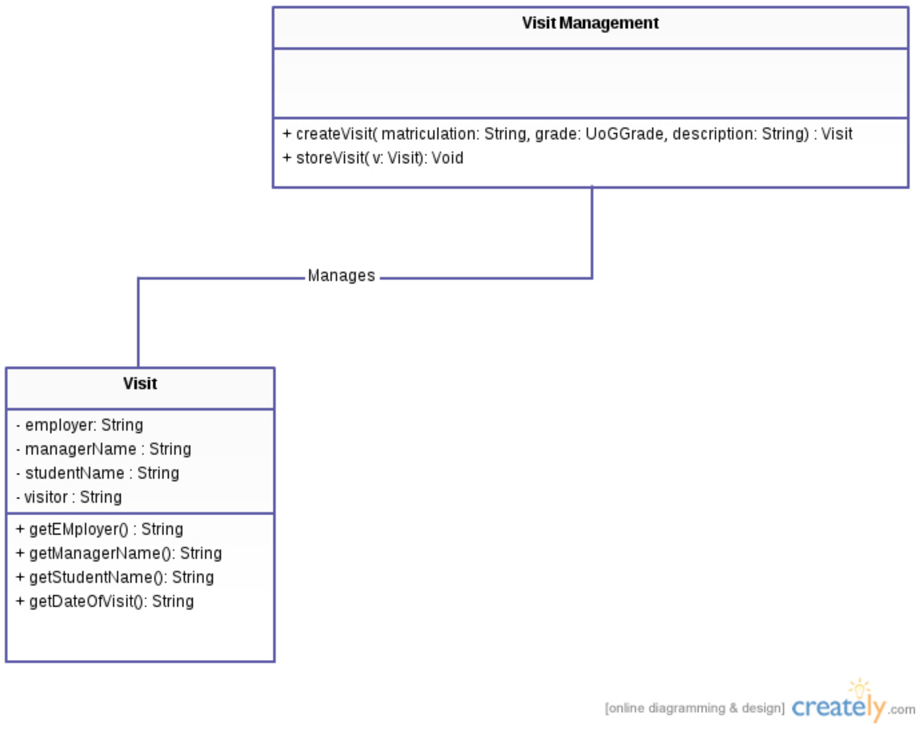


4.9 User Management

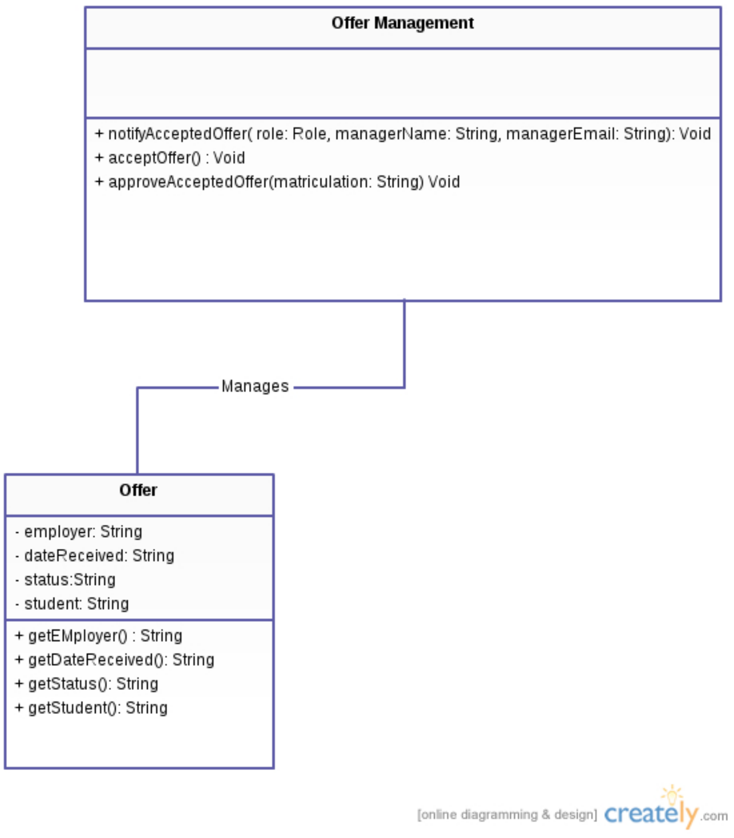


UserManagement
+ login(userName: String, password: String) : Bool
+ logout() : Void
+ getCurrentUser(): User
+ getCurrentEmployer(): Employer
+ getUsers(filter: Map): Map
+ registerNewEmployer(name: String, emailAddress:String)
+ getStudent(matric: String) : Student)
<<uses>>
<<uses>>
GUuser
- GUuser : GUuser[]
+ getGUuser (gUuserID: Int): GUuser
+ removeGUuser (gUuserId : Int) : Void
+ gUuserExists (gUuserId: Int) : Bool
+ storeGUuser (gUuserId:Int) : Void
Employer
- name : String
- employerId: Int
- email : String
#password: String
+ getName(): String
+ getEmployerId(): Int
+ getEmail(): String
[online diagramming & design] creately.com

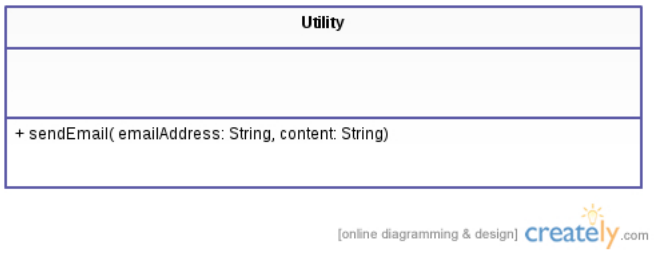
4.10 Visit Management


Visit Management
+ createVisit(matriculation: String, grade: UoGGrade, description: String) : Visit
+ storeVisit(v: Visit): Void
Manages
Visit
- employer: String
- managerName : String
- studentName : String
- visitor : String
+ getEMployer() : String
+ getManagerName(): String
+ getStudentName(): String
+ getDateOfVisit(): String
[online diagramming & design] creately.com

4.11 Offer Management


Offer Management
+ notifyAcceptedOffer(role: Role, managerName: String, managerEmail: String): Void
+ acceptOffer() : Void
+ approveAcceptedOffer(matriculation: String) Void
Manages
Offer
- employer: String
- dateReceived: String
- status:String
- student: String
+ getEMployer() : String
+ getDateReceived(): String
+ getStatus(): String
+ getStudent(): String
[online diagramming & design] creately.com

4.12 Utility


Utility
+ sendEmail(emailAddress: String, content: String)
[online diagramming & design] creately.com

5 API Specification

This section covers a detailed API specification for all the interfaces realised by the components of the internship management system.

5.1 Advert Store

Method: Public Advert getAdvert(integer advertId)
Description: Gets the advert with the given advert id.
Parameters: integer advertId

Return Type: Advert
Pre Condition: Advert with the given id is available within the system.
Post Condition:

Method:Public void removeAdvert(integer advertId)
Description: The advert with the given advert id is no longer available and removed from the system.
Parameters: Integer advertId
Return Type:
Pre Condition: Advert with the given id is available within the system.
Post Condition: Advert is no longer available on the system.

Method: Public Boolean advertExists(integer advertId)
Description: Checks if a specific advert does exist in the advert store.
Parameters: Integer advertId
Return Type: Boolean
Pre Condition:
Post Condition:

Method: Public void addAdvert(integer advertId)
Description: Adds a new advert with the given advert id to the advert store.
Parameters: Integer advertId
Return Type: void
Pre Condition: Advert is not already stored in the advert store.
Post Condition: Advert is now stored in the advert store.

Method: Public Map¡Integer, Advert¿ getAdverts()
Description: Gets all the adverts currently in the advert store.
Parameters:
Return Type: Map¡Integer, Advert¿
Pre Condition: Adverts stored in the advert store.
Post Condition:

5.2 Offer Store

Method: Public Offer getOffer(integer offerId)
Description: Gets the offer with the given offer id.
Parameters: Integer offerId
Return Type: Offer
Pre Condition: Offer with given id is stored in the system.
Post Condition:

Method: Public void removeOffer(integer offerId)
Description: The offer with the given offer id is removed from the system.
Parameters: Integer offerId
Return Type: void
Pre Condition: offer with the given id is available within the system.

Post Condition: offer is no longer stored on the system.

Method: Public Boolean offerExists(integer offerId)
Description: Checks if a specific offer is stored within the offer store.
Parameters: Integer offerId
Return Type: Boolean
Pre Condition:
Post Condition:

Method: Public void addOffer(Offer offerId)
Description: Adds a new offer with the given offer id to the offer store.
Parameters: Offer offerId
Return Type: void
Pre Condition: offer is not already stored in the offer store.
Pre Condition: student notifies course coordinator of internship placement offer.
Post Condition: offer is now stored in the offer store.

Method: Public Map¡Integer, Offer¿ getOffers()
Description: Gets all offers currently stored in the offer store.
Parameters:
Return Type: Map¡Integer, Offer¿
Pre Condition: Offers stored in the Offer store.
Post Condition:

5.3 Visit Store

Method: Public Visit getVisit(integer visitId)
Description: Gets the visit with the given visit id.
Parameters: Integer visitId
Return Type: Visit
Pre Condition: Visit details are stored in the system.
Post Condition:

Method: Public void removeVisit(integer advertId)
Description: The visit details with the given visit id removed from the system.
Parameters: Integer visitId
Return Type: void
Pre Condition: visit details with the given id are available within the system.
Post Condition: visit details are no longer available on the system.

Method: Public Boolean visitExists(integer adverted)
Description: Checks if a specific visit does is stored in the visit store.
Parameters: Integer visitId
Return Type: Boolean
Pre Condition:
Post Condition:

Method: Public void storeVisit(integer visitId)
Description: adds the details of a new visit with the given visit id to the visit store.
Parameters: Integer visitId
Return Type: void
Pre Condition: Visit is not already stored in the visit store.
Post Condition: visit is now stored in the visit store.

5.4 Employer Store

Method: Public Employer getEmployer(integer employerId)
Description: Gets the employer with the given employer id.
Parameters: Integer employerId
Return Type: Employer
Pre Condition: Employer with given employerId is available in the system.
Post Condition:

Method: Public boolean employerExists(integer employerId)
Description: checks if a specific employer is stored within the employer store.
Parameters: Integer employerId
Return Type: Boolean
Pre Condition:
Post Condition:

Method: Public void storeEmployer(integer employerId)
Description: Adds a new Employer with the given Employer id to the Employer store.
Parameters: Integer employerId
Return Type: void
Pre Condition: employer with the specified id does not exist in the employer store.
Post Condition: Employer is stored in the employer store.

5.5 GU User Store

Method: Public GUuser getGUuser(integer GUuserId)
Description: Gets the employer with the given GUID.
Parameters: Integer GUuserId
Return Type: GUuser
Pre Condition: GUuser with given GUuserId is available in the system.
Post Condition:

Method: Public boolean GUuserExists(integer GuuserId)
Description: Checks if a specific GU user is stored within the GU user store.
Parameters: Integer GUuserId
Return Type: Boolean
Pre Condition:

Post Condition:

Method: Public void storeGUuser (integer GUuserId)
Description: Adds a new GU user with the given GUuser id to the GU user store.
Parameters: Integer GUuserId
Return Type: void
Pre Condition: No user with the given id is stored in the GU user store
Post Condition: GU user is stored in the GU user store.

5.6 Employer Authentication

Method: Public void authenticateEmpUser(String username, String password)
Description: Checks that the username and password for the user are correct.
Parameters: String username, String password
Return Type: void
Pre Condition: Employer user is not logged in to the system.
Post Condition: Employer is logged into the system.

5.7 GU User Authentication

Method: Public void authenticateGUuser(String username, String password)
Description: Checks that the username and password for the user are correct.
Parameters: String username, String password
Return Type: void
Pre Condition: GU user is not logged in to the system.
Post Condition: GU user is logged into the system.

5.8 Advert Management Component

Method: public Advert getAdvert(Integer advertisementIndex)
Description: Returns advert specified by its index
Parameters: Integer advertisementIndex
Return Type: Advert
Pre Condition: Advert must be stored in system
Post Condition:

Method: public Map getAdverts(Map filter)
Description: Returns set of adverts, filtered by mapped options
Parameters: Map filter
Return Type: Map
Pre Condition: At least one advert stored in system
Post Condition:

Method: public Advert createAdvert(Map advertDetails)
Description: Creates & returns advert
Parameters: Map advertDetails
Return Type: Advert
Pre Condition:
Post Condition:

Method: public void removeAdvert(Integer advertIndex)
Description: Removes advertisment from system
Parameters: Integer advertIndex
Return Type: void
Pre Condition: Advertisement must be in system
Post Condition: Advertisement no longer present in system

Method: public void storeAdvertisement(Advert a)
Description: Stores advertisement in system
Parameters: Advert a
Return Type: void
Pre Condition:
Post Condition: Advert stored in system

Method: public void publishAdvertisement (Integer advertIndex, String comment)
Description: Makes advertisement visible to students & sends comment to employer by E-Mail
Parameters: Integer advertIndex, String comment
Return Type: void
Pre Condition: Advertisement must be in system & be unpublished
Post Condition: Advertisment is now visible to students

Method: public Role selectRole(Integer advertIndex, Integer roleIndex)
Description: Returns role for an advertisement, both specified by index
Parameters: Integer advertIndex, Integer roleIndex
Return Type: Role
Pre Condition: Advert & role must be in system
Post Condition:

5.9 User Management Component

Method: public boolean login(String userName, String password)
Description: Logs user into the system
Parameters: String userName, String password
Return Type: boolean
Pre Condition: User exists on system & not currently logged in
Post Condition: Set as current user

Method: public Map getUsers(Map filter)
Description: Returns set of users, filtered by mapped options

Parameters: Map filter
Return Type: Map
Pre Condition: At least one user in system
Post Condition:

Method: public Employer registerNewEmployee(String name, String emailAddress)
Description: Returns employer after setting up an account for them
Parameters: String name, String emailAddress
Return Type: Employer
Pre Condition: Employer shouldn't exist on system
Post Condition: Employer has account on system

Method: public void notifyAcceptedOffer(Role role, String managerName, String managerEmail)
Description: Notifies coordinator of accepted offer by E-Mail
Parameters: Role role, String managerName, String managerEmail
Return Type: void
Pre Condition: Student must be logged in
Post Condition:

5.10 Offer Management

Method: public void acceptOffer(Integer OfferId)
Description: Sets the status for the offer from pending to accepted.
Parameters: Integer OfferId
Return Type: void
Pre Condition: Offer is stored in the offer store.
Post Condition: Status for offer is now 'Accepted'.

Method: public void approveAcceptedOffer(String matriculation)
Description: Approves offer most recently accepted by student with this matriculation id
Parameters: String matriculation
Return Type: void
Pre Condition: Student needs to have accepted at least one offer
Post Condition:

Method: public Offer getOffer(Integer OfferID)
Description: Gets offer with the given offer ID.
Parameters: Integer OfferID
Return Type: Offer
Pre Condition: Offer is stored in the Offer store.
Post Condition:

5.11 Visit Management

Method: public Visit createVisit(String matriculation, UoGGrade grade, String description)
Description: Creates and returns a visit
Parameters: String matriculation, UoGGrade grade, String description
Return Type: Visit
Pre Condition: Student must exist on system
Post Condition:

Method: public void storeVisit(Visit v)
Description: Stores visit in system
Parameters: Visit v
Return Type: void
Pre Condition:
Post Condition:

6 Acceptance Tests

Before the Internship Management System can be deployed we must create several acceptance tests to ensure that the program covers all the requirements from the specification. These acceptance tests are not designed to identify bugs but to ensure that the user can interact with the application and use it's advertised functions.

Identifier	T.C.5.1.1
Use case	Login
Scenario	Primary
Set up	System initialised with default parameters - course coordinator username and password.
Includes	none
Procedure	Not defined yet
Inputs	User = ccadmin, password=pass
Outputs	User set from guest to ccadmin

Identifier	T.C.5.1.2
Use case	Login
Scenario	Invalid login/password for course coordinator
Set up	System initialised with default parameters - course coordinator username and password
Includes	none
Procedure	Not defined yet
Inputs	User = ccadmin, password=pass1
Outputs	User has not changed to ccadmin or gained access to admin features.

Identifier	T.C.5.1.3
Use case	Login
Scenario	Primary
Set up	System initialised with default parameters - student username and password
Includes	none
Procedure	not defined yet
Inputs	User = 2060267s, password=password
Outputs	User has changed to student

Identifier	T.C.5.1.4
Use case	Login
Scenario	Primary
Set up	System initialised with default parameters - employer username and password
Includes	T.C.5.5.1
Procedure	Not defined yet
Inputs	User = Google, password=Mypassword1998
Outputs	User has changed to company

The above Tests cover the different user types logging in with both invalid and valid passwords.

Identifier	T.C.5.2.1
Use case	getAdvertisements
Scenario	Primary - View approved adverts
Set up	logged in as a student
Includes	T.C.5.1.3 (logged in as student)
Procedure	Not defined yet
Inputs	view
Outputs	All published adverts are displayed

Identifier	T.C.5.2.2
Use case	getAdvertisements
Scenario	Primary - show all adverts, pending and published
Set up	System initialised with default parameters - logged in as course coordinator
Includes	T.C.5.1.1 (course coordinator), T.C.5.5.1(submitted ads)
Procedure	Not defined yet
Inputs	View pending
Outputs	All pending adverts are shown

Identifier	T.C.5.3.1
Use case	SubmitAdvert
Scenario	Primary
Set up	Logged in as a employer
Includes	T.C.5.1.4 (employer login)
Procedure	Not defined yet
Inputs	Internship details
Outputs	Email sent to course coordinator

Identifier	T.C.5.4.1
Use case	selectAdvertisements
Scenario	Primary
Set up	System initialised with default parameters - logged in as student
Includes	T.C.5.1.3, T.C.5.2.1
Procedure	Not defined yet
Inputs	Id for the advert to view
Outputs	Selected advert details are shown

Identifier	T.C.5.5.1
Use case	RegisterNewEemployer
Scenario	Primary
Set up	System initialised with default parameters - logged in as course coordinator
Includes	T.C.5.1.1
Procedure	Not defined yet
Inputs	Valid employer details - name and email address
Outputs	Account created for employer

Identifier	T.C.5.5.2
Use case	RegisterNewEmployer
Scenario	Invalid details
Set up	System initialised with default parameters - logged in as course coordinator
Includes	T.C.5.1.1
Procedure	Not defined yet
Inputs	Invalid/missing employer details
Outputs	Warning message is shown with error details

FIX...

Identifier	T.C.5.6.1
Use case	ViewAdvertisements
Scenario	Primary
Set up	System initialised with published advertisements
Includes	T.C.5.7.1, T.C.5.1.1
Procedure	Not defined yet
Inputs	View ads
Outputs	Advertisement summary is shown

Identifier	T.C.5.7.1
Use case	publishAdvert
Scenario	Primary
Set up	System initialised with default parameters - logged in as course coordinator
Includes	T.C.5.1.1, T.C.5.3.1
Procedure	Not defined yet
Inputs	advertisement ID, comment
Outputs	Advert is published and feedback sent to employer (email)

FIX....

Identifier	T.C.5.7.2
Use case	publishAdvert
Scenario	Advert is not suitable for students
Set up	System initialised with default parameters - logged in as course coordinator
Includes	T.C.5.1.1, T.C.5.3.1
Procedure	Not defined yet
Inputs	CHANGE
Outputs	Advert is not published, employer is notified via email

Identifier	T.C.5.8.1
Use case	NotifyAcceptedOffer
Scenario	primary
Set up	Offer details are filled in, logged as a student
Includes	T.C.5.1.3
Procedure	Not defined yet
Inputs	Role, manager name, manager email address
Outputs	Email sent to course coordinator for approval

FIX....

Identifier	T.C.5.8.2
Use case	Notify course coordinator
Scenario	student supplies incorrect offer details
Set up	logged as a student
Includes	T.C.5.1.3
Procedure	Not defined yet
Inputs	Incorrect role details
Outputs	Warning message is shown

Identifier	T.C.5.9.1
Use case	ApproveAcceptedOffer
Scenario	Primary
Set up	logged as a course coordinator
Includes	T.C.5.1.1, T.C.5.8.1
Procedure	Not defined yet
Inputs	students matriculation number
Outputs	Offer status is now approved in student details

FIX....

Identifier	T.C.5.9.2
Use case	ApproveAcceptedOffer
Scenario	Offer is not approved.
Set up	logged as a course coordinator
Includes	T.C.5.1.1, T.C.5.8.1
Procedure	Not defined yet
Inputs	Students matriculation number
Outputs	Offer status is not set to approved in student details

Identifier	T.C.5.10.1
Use case	selectStudent
Scenario	Primary
Set up	logged as a course coordinator
Includes	T.C.5.1.1
Procedure	Not defined yet
Inputs	Students matriculation number
Outputs	student details - name, matriculatuion numebr and program

Identifier	T.C.5.10.2
Use case	selectStudent
Scenario	matriculation number is incorrect
Set up	logged as a course coordinator
Includes	T.C.5.1.1
Procedure	Not defined yet
Inputs	Students matriculation number
Outputs	Students details not shown.

SubsectionTest Plan Implementation
These test cases will be implemented using the JUnit test harness within Java.