University of Glasgow | Department of Computing Science

D6: SYSTEM DESIGN DOCUMENT

Deliverable ID	D6
Deliverable Title	System Design Document
Project	PSD3 Group Exercise 2
Team	L
Authors	Dan Tomosoiu Peeranat Fupongsiripan Hector Grebbell Michael Kilian Tony Lau
Deliverable Date	30 Jan 2013
File Name	SystemDesign.tex
Version	1.0

1 Introduction

1.1 Identification

This document outlines the proposed design for the Internship Management System.

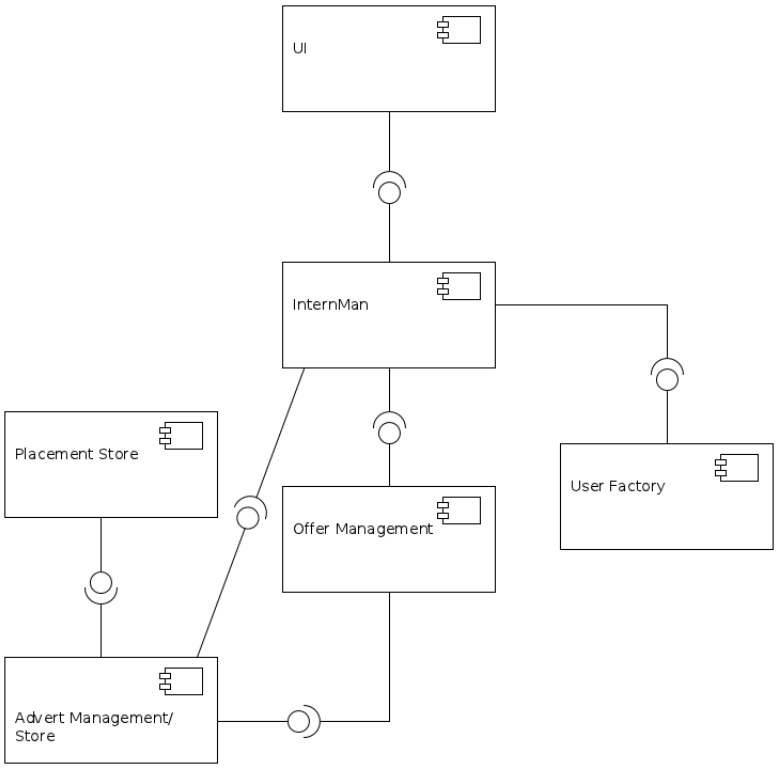
1.2 Related Documentation

- Requirements Specification Document

1.3 Document Status and Schedule

30/01/2013 - First draft

2 System Overview



The proposed system consists of the following components:

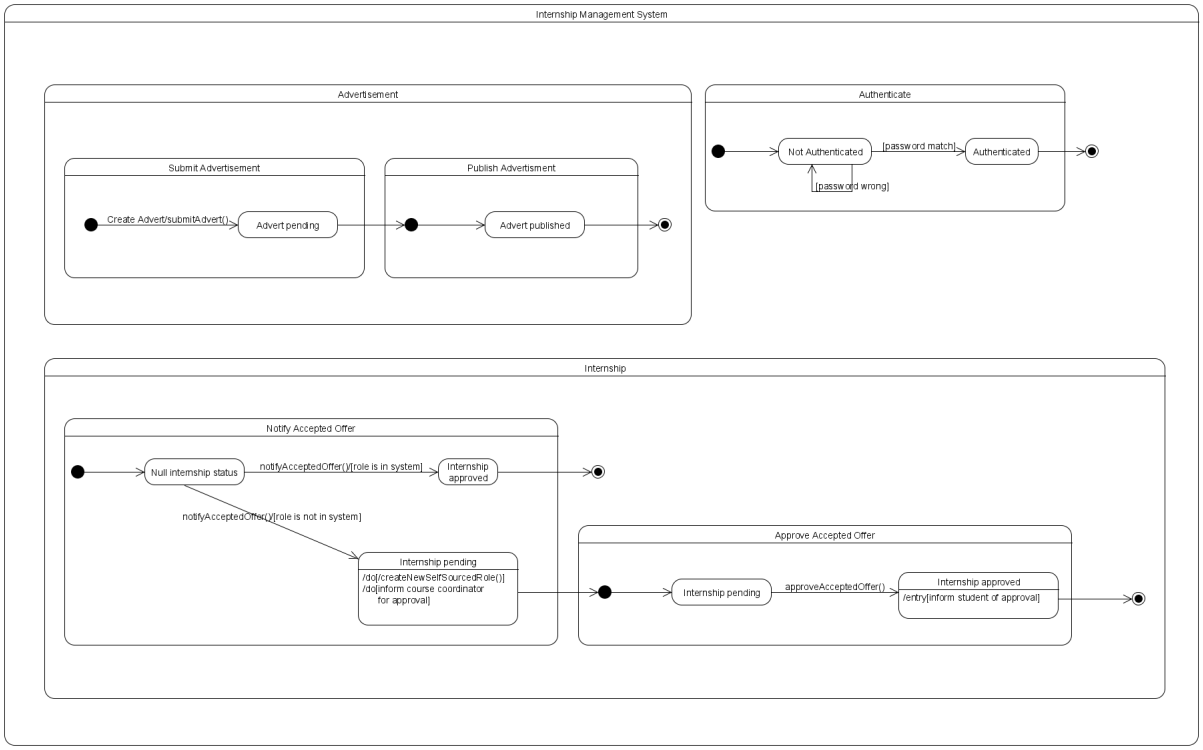
- InternMan: This component contains an implementation of the InternMan facade provided and can be seen as the 'main' component for the system.
- User Factory: Provides methods for creating new users of different types, storing created users and fetching users from this store. This component also provides services for authenticating a user's login credentials using a stored username/password combination.
- Offer Management: Handles offers made to students.

- Advert Management/Store: This component provides the necessary services to create and view adverts, associated these with an employer account, etc. It simultaneously manages the storage and retrieval of adverts from an appropriate source.
- Placement Store: Provides services for manipulating internships and associating these with an approriate role.

3 State Machine Diagram

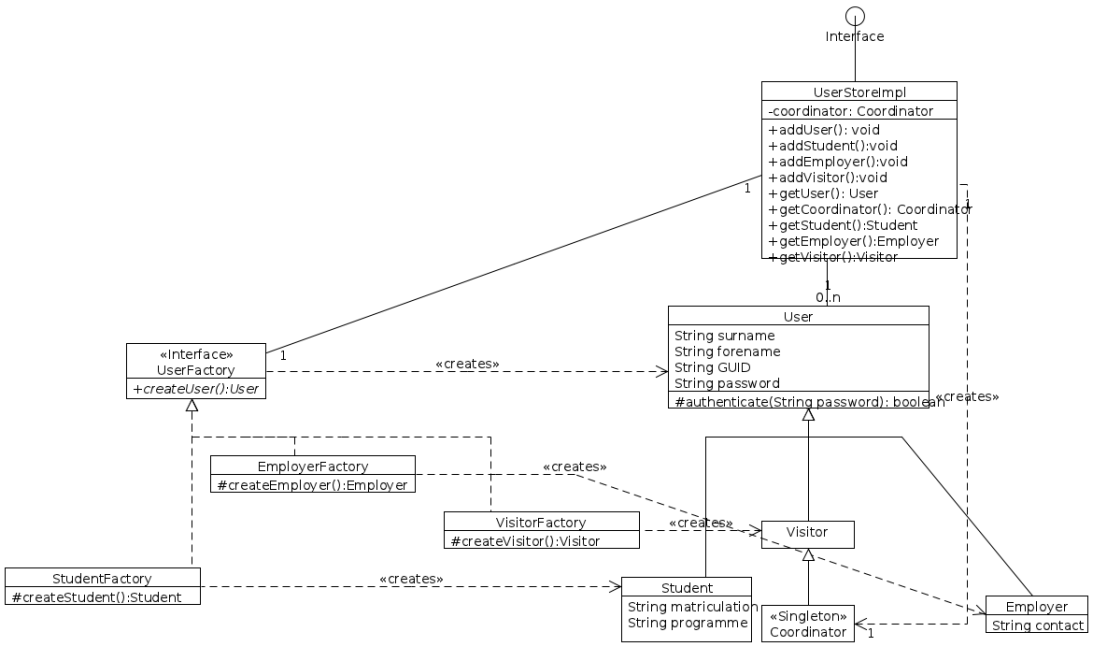
3.1 Description and Rationale

The state machine diagram shows the state of objects within the Internship Management System at any given moment in time.



4 Component Breakdown

4.1 User Factory



4.1.1 Description and Rationale

There is a need to create many different categories of user in the system who share attributes and methods. For example all users must be able to log in using a method to authenticate an input password. In this situation it is appropriate to employ the factory design pattern for the creation of generic users, visitors, students and employers. The coordinator is not instantiated by a factory, but instead is implemented according to the singleton design pattern.
UserStoreImpl groups the methods of these factories and entities together to provide methods for creating and fetching each class of user. For the moment the exact means by which these users will be stored has not been decided, but this should also be handled by UserStoreImpl. The method set for UserStoreImpl is equivalent to the API for the UserFactory component.

4.1.2 API

UserStoreImpl implements UserStore
Methods
public void addUser(String surname, String forename, String GUID, String password)
Adds a new user to the store.
Parameters
String surname
String forename
String GUID
String password

public void addVisitor(String surname,String forename,String GUID, String password)
Adds a new visitor to the store
Parameters
String surname
String forename
String GUID
String password

public void addVisitor(String surname,String forename,String GUID, String password,String matriculation,String programme)
Adds a new student to the store
Parameters
String surname
String forename
String GUID
String password
String matricaulation
String programme

public void addVisitor(String surname,String forename,String GUID, String password,String contact)
Adds a new employer to the store
Parameters
String surname
String forename
String GUID
String password
String contact

public User getUser(String GUID, String password)
Returns a user specified by the GUID, if authentication is successful.
Return the user specified by GUID, if the password is a match
Parameters
String GUID
String password

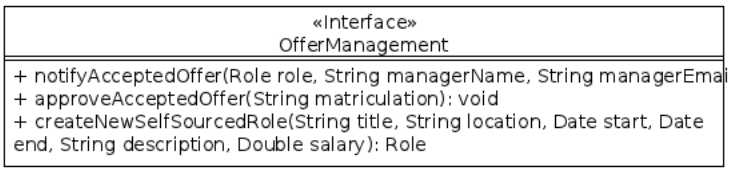
public Visitor getVisitor(String GUID, String password)
Returns a visitor specified by the GUID, if authentication is successful.
Return the visitor specified by GUID, if the password is a match
Parameters
String GUID
String password

public Student getVisitor(String GUID, String password)
Returns a Student specified by the GUID, if authentication is successful.
Return the student specified by GUID, if the password is a match

Parameters
String GUID
String password

public Visitor getVisitor(String name, String password)
Returns a visitor specified by the name if authentication is successful.
Return the user specified by name, if the password is a match
Parameters
String name
String password

4.2 Offer Management



4.2.1 Description and Rationale

This component is needed to allow students to notify the course coordinator that they have accepted an internship. If the internship was advertised in the internship management system, then it is automatically approved. Otherwise, the student can create a new self-sourced role which will send an email to the course coordinator asking for approval. Approval is done through this component.

4.2.2 API

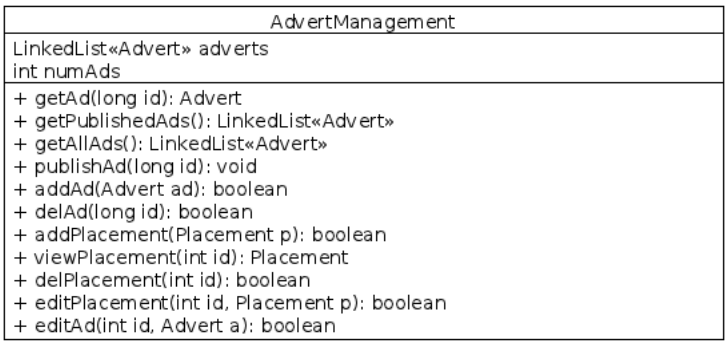
OfferMangementImpl implements OfferManagement
Methods
public void notifyAcceptedOffer(Role role, String managerName, String managerEmail)
Sends email to coordinator informing that a student has accepted an internship
Parameters
Role role
String managerName
String managerEmail

public void approveAcceptedOffer(String matriculation)
Sends email to coordinator informing that a student has accepted an internship
Parameters
String matriculation

public Role createNewSelfSourcedRole(String title, String location, Data start, Date end, String description, Double salary)

Sends email to coordinator informing that a student has accepted an internship
Return
Returns the newly created self-sourced role
Parameters
String title
String location
Data start
Date end
String description
Double salary

4.3 Advert Management/Store



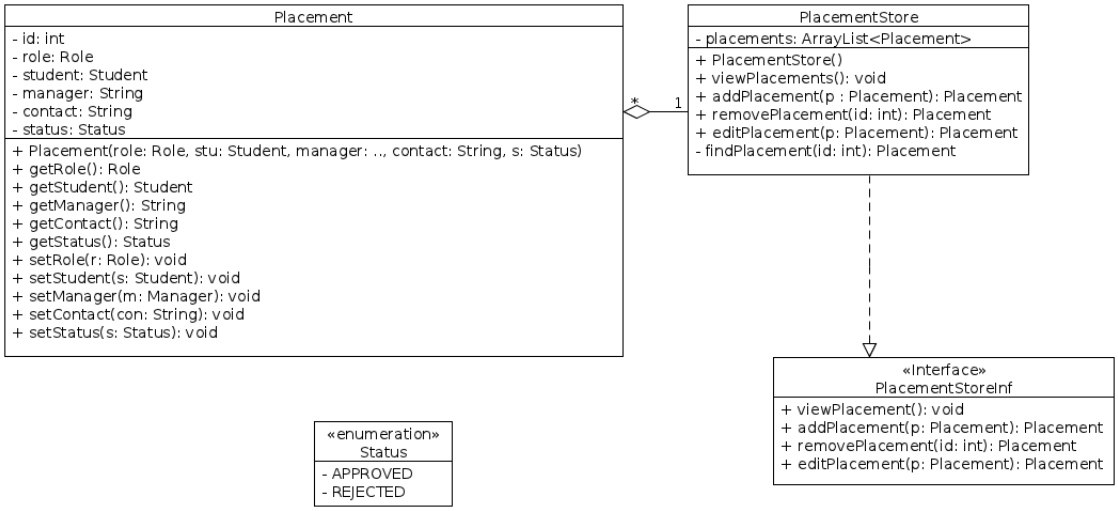
4.3.1 Description and Rationale

Advert Management/Store is needed to handle creation, selection and publishing of all advertisements within the system. It interacts with other components which need information on adverts through the selectAdvertisement method which provides a handle on an advert with a supplied index.

4.3.2 API

TODO

4.4 Placement Store



4.4.1 Description and Rationale

Placement Store contains all information on placements in the system at any given time and is needed to allow other componenets to keep track of which placements are in the system and their status.

4.4.2 API

Interface PlacementStoreInf

Methods
public void viewPlacements()
Prints out all existing placements.

public Placement addPlacement(Placement p)
Add new placement.
Return: Placement p added if successful, otherwise null.
Parameters
Placement p

public Placement removePlacement(int id)
Return: Placement with identification number id removed if successful , otherwise null.
Parameters
int id

public Placement editPlacement(Placement p)
Edits existing placement.
Return: Placement edited if successful, null otherwise.
Parameters
Placement p

Class Placement
Contains details about the internship including the student taking and role involved.

public Placement(Role role, Student student, String manager , String contact, Status status)
Parameters
Role role
Student student
String manager
String contact
Status status

public Role getRole()
Return: Role of placement if exists, null otherwise.

public Student getStudent()
Return: Student secured the placement , null otherwise.

public String getManager()
Return: Manager if exists, null otherwise.

public String getContact()
Return: Contact details, if exists, null otherwise.

public Status getStatus()
Return: returns the status of a particular placement (value can be either APPROVED OR REJECTED)

setRole(Role r)
Return:
Parameters
Role r

setStudent(Student s)
Return:
Parameters
Student s

setManager(String m)
Return:
Parameters
String m
setContact(String c)
Return: Parameters String c

setStatus(Status s)
Return:

Parameters
Status s

A Glossary

- UML - Unified Modelling Language
- API - Application Programming Interface
- PSD/PSD3 - Professional Software Development (3)