IN2013: Object – Oriented Analysis and Design

Tutorial 3: Identifying classes, attributes and operations

Model Answer

This tutorial is focused on identification of the problem domain classes for BAPERS (the scenario is provided in the Appendix) and crating the 1st cut class diagram.

The assignment is designed to take longer than an hour to complete. I would like you to try your best in class but be aware that it is likely that you may need to spend extra effort to complete the assignment at home, after the end of the Tutorial session.

You might start with a pencil and paper solution and proceed to entering the diagram into Visual Paradigm after obtaining a good sketch of what you consider a correct solution.

You can work alone or form a small team of 2-3 people with your colleagues from the same tutorial group. Group discussions of the possible solutions while you produce them, will bring additional benefits. Each member of the team must strive to develop their own solution.

Exercise 1:

Apply the noun/verb analysis to the scenario and identify the list of problem domain classes with their attributes and important operations.

Model answer:

The text is subjected to noun-verb analysis. First the noun phases and highlighted with green and the verb phrases with light-blue.

The case study describes the requirements for automating the processes at the photographic laboratory 'The Lab' operated by Bloomsbury's Image Processing Laboratory (**BIPL)**. **BIPL** handles the work of professional photographers and carries out *jobs* on behalf of customers, but the customers do not use the system. The access to the system is limited to the laboratory staff only.

Each job is given a *unique identifier* and assigned *urgency* (a job may be urgent or normal which affects the turnaround time) when the job is placed. Urgent jobs must be completed within 6 hours while the normal jobs – within 24 hours. A job may consist of any number of the *standard tasks* offered by BIPL.

Currently there are around 30 standard tasks. Each task has an identifier (e.g., a number), is carried out at a particular location within the laboratory, has a quoted price, and an (estimated) duration. Some of the tasks are shown in Table 1 below. The case study does not require that the list of tasks be maintained (i.e. BAPERS does NOT offer functionality for adding new tasks, removing, or updating existing tasks).

Table 1

Task ID	Task Description	Location	Price (£)	Duration (min)
1.	Use of large copy camera	Copy Room	19.00	120
2.	Black and white film processing	Development Area	49.50	60
3.	Bag up	Packing Department	6.00	30
4.	Colour film processing	Development Area	80.00	90

5.	Colour Transparency processing	Development Area	110.30	180
6.	Use of small copy camera	Copy Room	8.30	75
7.	Mount Transparencies	Finishing Room	55.50	45
…	Etc.	Etc.	Etc.	Etc.

At any given time, hundreds of jobs will be in progress or pending within the laboratory. Every job accepted must be chargeable to a valid *customer account*, either an existing account, or a newly created account (e.g., a photographer may walk in, leave a film to be developed, and pay cash on collecting the finished job).

BIPL want to enable the employee on the reception desk to enter the job on a computer terminal. The material will be labelled with the job number and taken down to the laboratory.

The laboratory staff will interrogate **BAPERS** to ascertain the tasks required. As the job is transferred from one location to another in the laboratory, the staff responsible for each task will record its completion on a computer terminal in their location before passing it on. A terminal will be required in each of the following locations: Copy Room, Dark Room, Development Area, Printing Room, Finishing Room, and Packing Department.

Many jobs will be going through the laboratory at any given time, and confusion between them must be avoided. At all cost, loss or mistreatment of the customer's material must not occur. Queues of work may build up at the processing stations, but flexible scheduling is required to allow priority to be given to *urgent* jobs over the *regular/normal* jobs. The system should provide functionality for inspecting the list of active/pending jobs as well as already completed ones, including the inspection of the progress of individual tasks (active/pending tasks vs. completed ones).

BAPERS must therefore provide the following main facilities:

BAP-ACCT *Accept job at reception*: Identify existing (or create new) customer account (name, phone). Assign job number. Record the deadline for completion of the job. This functionality will be mainly used by the receptionists but will also be made accessible by either Office manager or Shift manager in case receptionist(s) are absent.

BAP-PROC *Process a job through laboratory*: Respond to enquiries from any computer terminal about status of any jobs in progress, or of all jobs (including the completed ones). Update status of any given job by recording completion of current task and commencement of next (possibly with transfer of material to a new location). This functionality is available to Technicians, Shift and Office Manager.

In addition, alert Shift and/or Office manager (by, for example, displaying a visual cue with appropriate text) if the expected time to complete outstanding tasks for any job is likely to exceed the set time period, i.e. if the deadline for the job is not likely to be met; the alerts should be performed only for these two user roles.

BAP-PAYM *Payment processing*. The customers are supposed to pay once the jobs they had placed have been completed. Customers can pay by cash or credit/debit card only. The system is connected to an external payment processor to move funds between the customers' credit/debit cards and the BIPL bank account. In case of a card payment BAPERS is expected to connect to the external payment processor and on successful completion of the card payment the payment is recorded with the

following details: card payment, type of card, and the last 4 digits of the card used. Cash payments are recorded, too. Several jobs can be paid at once. Only payment in full is accepted (i.e. no partial payments are allowed). This functionality is available to Receptionist, Shift and Office Manager.

BAP-ADMN *Administering the system.* This includes creating a user account for BAPERS and setting up access privileges. The following user roles must be implemented: Office manager, Shift manager, Receptionist and Technician. This functionality will be available to Office Manager only.

*A consolidated list of noun phrases is provided below with commentary about whether the noun (phrase) is a good candidate **class**, **attribute** or simply do not belong to the problem domain. I highlight the discovered problem domain classes in green. The attributes are highlighted with light blue.*

- *Case study – not part of the problem domain*
- *Requirements - not part of the problem domain*
- *Processes - not part of the problem domain*
- *Laboratory – part of the problem domain, but is not modelled, as the lab is the organization for which the system (i.e., BAPERS) is developed. Should the system become a general-purpose software, the laboratory (more generally the organization using the software) may be modeled as, the "business", with relevant attributes, e.g., organisation's logo etc. For the time being (of BAPERS being a bespoke software system) we just assume that the laboratory is not modeled.*
- *Work of photographs (material, film) – this noun phrase becomes eventually a job (see below) and is part of the problem domain.*
- *Job (jobs) – an essential problem domain class.*
- *Customer(s), Photographer – these nouns will lead to CustomerAccount (see below)*
- *System – this defines the system being modeled, will appear as the subject in the use-case model. With respect to class diagram, this noun refers to the entire class model and therefore is ignored.*
- *Laboratory staff (employee) – this is an essential concept, and we could record staff as a class. On the one hand, staff are **actors** and therefore are not part of the system. On the other hand, staff must have **UserAccount** (see below), which will store the login details to make it possible for them to login. UserAccount is a candidate for a problem domain class, which may be further specified via inheritance. Whether or not inheritance is needed will depend on the associations with other classes, which will transpire when we model the roles of the classes.*
- *Task(s) – an essential problem domain class, which will have a set of attributes.*
- *Task description – an essential class TaskDescription with a set of attributes as defined by Table 1 (ID, description, price, location and duration).*
- *(Task) identifier – an attribute of TaskDescriptor (id).*
- *Location – an attribute of TaskDescriptor (as in Table 1).*
- *(Task) price – an attribute of TaskDescriptor (as in Table 1).*
- *List of tasks – refers to a **collection** of tasks. Lists are ways of grouping many objects of the same type (Task) and dealing with them efficiently at run time. Lists are modelled using collections classes such as Vector/Array classes. Such classes, however, are not part of the problem domain, but part of the so-called solution domain. Classes from the solution domain are not used in analysis. They are only added to class diagrams when we refine the analysis class diagram into a design class diagram. In analysis class diagram only captures the problem domain classes. For this reason, we will capture the existence of lists of Tasks via the multiplicity of the associations that the class Taks may have with other classes, e.g., with the class Jab.*
- *Duration – an attribute of TaskDescriptor (as in Table 1).*
- *Time – ignored. A generic term, not part of the problem domain.*
- *CustomerAccount – a problem domain class. We need a set of attributes (Name, PhoneNumber, Email) to define the CustomerAccount. Importantly, these accounts are not user accounts, but accounts on which we store data on the customers of the Lab.*

- *Reception desk – a term qualifying the BAPERS users who might be taking in new jobs. Not explicitly modeled in class diagram.*
- *Computer terminal – not part of the problem domain. Defines loosely a boundary class, which we will eventually add (when doing robustness analysis).*
- *(task/job) completion – refers to a moment in time. Not explicitly modeled in class diagram.*
- *CopyRoom, DarkRoom, DevelopmentArea, PrintingRoom, FinishingRoom, PackingDepartment – values taken by the attribute Location of class Task. May be modeled as enumeration (typically done in design) or are ignored in analysis.*
- *Confusion – ignored, outside the problem domain.*
- *Loss, Mistreatment – ignored. These refer to non-functional requirements (possibly about reliability), not part of the problem domain.*
- *Queues of work (List of jobs, List of Active/pending jobs). – ignored. Similarly, to the list of tasks, this will be addressed in detail in design. In analysis we will use multiplicity to capture the concept of multiple jobs being in processing.*
- *Stations – ignored. Refers to the equipment used to process tasks. Not modeled in analysis class diagram.*
- *Scheduling – ignored. refers to ways of changing the order in which jobs are processed. This refers to a piece of functionality specific to a list of jobs. Since we chose not to model the list of tasks beyond multiplicity, this aspect will be postponed until design (when lists of jobs/tasks will be added to the refined class diagram).*
- *Priority – an attribute of class Job (with values taken from the enumeration set: {urgent, normal}).*
- *Progress (status) of task – an attribute of Task with values taken from the set (pending, active, completed).*
- *Facilities – ignored, a generic term referring to the types of functionality in BAPERS.*
- *Deadline (time to complete, set time period) – an attribute of Job.*
- *Receptionist, Office Manager, Shift Manager, Technician – types of users. Modeled via inheritance from UserAccount. Since the only difference between the user accounts are the associations with other classes, it is better to use role attached to associations to indicate the types of user.*
- *Enquiries – this refers to Viewing jobs functionality. Unless the enquiries are logged, i.e., their details (e.g., who, when, what) are recorded, this concept is not captured in analysis class diagram.*
- *(Job) Status – an attribute of Job. Values (pending, active, completed). Active – when at least one task has become active/competed and at least one has not been completed. This attribute is an example of a derived/computed attribute (i.e., is a function of the values of other attributes). UML has a specific syntax for it.*
- *Transfer (of material) – this refers to moving the material from one location to another. We record the Location of the material via the attribute 'location' of Job.*
- *Alert(s) (visual cue) – a class Alert. Must be associated with a Job. Implicitly, must be also associated with the Managers (Office and Shift) as they are the only ones to receive instances of Alert.*
- *Text (of visual cue) – an attribute of Alert.*
- *Role (privileges) – refers to the types of users, but also an attribute of UserAccount. The model developed in the model answer uses roles attached to associations of UserAccount to capture the roles of the different users.*
    - *We may infer from the references to user roles, that some sort of Rale Based Access Control (RBAC) is used and infer that:*
        - *Username – attribute of UserAccount*
        - *Password – attribute of UserAccount*
- *Cash (payment) – a type of payment, modeled as CashPayment, a sub-class of Payment.*

- *Card (payment) – a type of payment, modeled as CardPayment a sub-class of payment.*
    - *Payment is a generalization of CashPayment and CardPayment. Payment must be associated with 1…* Jobs, as the case study states that several jobs can be paid for at once.*
- *Payment processor – a secondary actor, **not modeled** in analysis class diagram. In the next tutorial a boundary class will be added to allow BAPERS to communicate with the payment processor.*
- *Funds – an attribute of Payment (amount).*
- *CreditDebitCard – a problem domain class. May be merged with CardPayment.*
- *Bank account – an attribute of BIPL, which is not modeled in analysis.*
- *Card Type – an attribute of CreditDebitCard (or of CardPayment if merged with CreditDebitCard).*
- *4 digits - an attribute of CreditDebitCard (or of CardPayment if merged with CreditDebitCard).*
- *user account – a problem domain class UserAccount, see the RBAC discussion above.*

Consolidated list of verbs. The verbs can refer to class operations, named associations or roles. Generic verbs are ignored. I marked the discovered operation with green and the associations/roles with light blue.

- *Describes – ignored. This is a generic verb, not part of the problem domain.*
- *Operated – ignored. This is a generic verb, not part of the problem domain.*
- *Handles – ignored. This is a generic verb, not part of the problem domain.*
- *Carries out – ignored. This is a generic verb, not part of the problem domain.*
- *Is limited – ignored. This is a statement about the actor of BAPERS. No modeling consequences.*
- *Is given (unique id) – an operation of Job assignJobID(). Since the ID is not changed once the object is created, the operation may be unnecessary: the ID can be assigned at the time the object is instantiated (i.e., by the constructor).*
- *Assigned (urgency) – operation of Job changePriority(). Omitting this operation would be wrong. The urgency may be changed at a later stage, hence having it in the model is necessary.*
- *May be – refers to the possible values that urgency may take. A constraint on the values of urgency.*
- *Is placed – ignored. There are no modeling consequences.*
- *Must be completed – defines a rule for computing the default value of job completion time.*
- *May consist of – named association <consists of> between Job and Task classes.*
- *Offered – ignored. A general verb, no modeling consequences.*
- *Are – reference to the size of enumeration StandardTasks (this will be added in design, not explicitly modeled in analysis).*
- *Carried out at – could be captured as a named association between task and location (in case location is modeled as a separate class). In the class diagram location is modeled as an attribute of task, hence <carried out at> as an association is not explicitly modeled.*
- *Has – refers to the attribute price of Task. No modeling consequences (as the attribute has already been captured above).*
- *Shown – ignored, no modeling consequences.*
- *Require – ignored, no modeling consequences. Refers to limiting the scope of the case study.*
- *Be maintained – ignored, no modeling consequences (limits the scope of the cases study)*
- *Offer – ignored, a generic term, no modeling consequences.*
- *Adding (new task) – this refers to managing the list of standard tasks, which is not required. However, it may also be treated as an operation addTask() of job (tasks get added to a job). Will trigger instantiating a new task.*
- *Removing, Updating tasks – ditto. Imply useful operations, removeTask() and updateTask() for job.*
- *Will be in progress, pending – refer to the possible states of a job. Retrieving the job state could be done by introducing an operation retriveJobState() for Job.*
- *(must be) chargeable – named association between Job and CustomerAccount.*

- *Walk in, leave – ignored, generic verbs, no modeling consequences.*
- *To be developed – ignored, no modeling consequences.*
- *Pay – could be captured as a named association between classes CustomerAccount and Job. However, such an association may be obsolete as the association between Payment and Job already implies who should make the payment (the CustomerAccount associated with the Job).*
- *Collecting – ignored, refers to an activity outside the scope of BAPERS. BAPERS does not deal with delivery/collecting the materials from the lab. No modeling consequences.*
- *(Want) to enable – ignored, a generic verb, no modeling consequences.*
- *Enter (the job) – a named association between the UserAccount (who placed the job) and the Job. This name, however, is not used in the diagram, as roles are used with the association. Using association names and roles would make the model unnecessarily complicated. The rule of thumb is that for associations with roles, the association name is omitted.*
- *Will be labeled – ignored, this is outside BAPERS. Whether the printed labels will be attached to material or not is outside BAPERS (no record is made if and who attached the label).*
- *Taken down – ignored, again, outside BAPERS. The system just assumes that the Receptionist (or the Manager working at the desk) will do what they are supposed to do, but whether this is the case is not tracked.*
- *Interrogate – a possible operation of ListofTasks to retrieve the list. Since in analysis we are not modeling the list explicitly, this operation is not used.*
- *To ascertain (the tasks required) – seems to refer to amending the job (adding, removing, updating tasks). The necessary operations have already been captured in the analysis above. No additional modeling consequences.*
- *Transferred (from location to location), passing on – refers to an operation setCurrentLocation() of Job.*
- *Will record (completion of task) – operation complete() of Task.*
- *Will be required – refers to deployment of terminals in the Lab. No modeling consequences for analysis class diagram. Useful for deployment diagram (which will be discussed a few weeks later).*
- *Will be going through (the laboratory) – ignored, a generic phrase, no modeling consequences. Seems to refer to the fact of many job instances being active simultaneously, which is not captured by the class diagram, but may be captured by object diagrams.*
- *(confusion) Must be avoided – ignored, reference to non-functional requirements, no modeling consequences.*
- *Must not occur – ditto.*
- *(queues) may build up – ignored, a reference to the possibility of many task instances pending for the same equipment. This is modeled with object diagrams, not in class diagram, which is the focus here.*
- *(scheduling) is required – ignored as irrelevant for problem domain. Could have been captured as a useful operation Schedule() of a class ListofJobs, which we chose not to model in analysis, but will be added in design. This operation schedule() can be defined for a control class used in analysis (in the tutorial next week).*
- *(priority) to be given – ignored, this is a rule to be used in developing scheduling algorithm of Schedule() operation, which is related to design.*
- *Should provide (functionality) – a generic phrase, no modeling consequences.*
- *Inspecting (the list of active pending jobs) – operation inspectJobs() of a control class (to be added next week).*
- *Including (the inspection of job tasks) – operation inspectTasks() of Job.*
- *Provide – a generic verb, no modeling consequences.*
- *Accept (job) – refers to the constructor of a Job.*
- *Identify (customer) – operation identifyCustomer() of a control class (to be added to the model in the next tutorial).*

- *Assign (job number) – operation assignJobID() of Job, used by the class constructor.*
- *Record (job completion deadline) – operation computeDeadline() of Job. Used by the class's constructor.*
- *Will be mainly used – role of Receptionist <creates> in the association between classes UserAccount and Job.*
- *made accessible – adds the same role, <create> for OfficeManager and ShiftManager.*
- *Are absent – ignored, a generic phrase with no modeling consequences.*
- *Process (a job) – generic term, no modeling consequences.*
- *Respond (to enquiries from terminal) – retriveJobStatus() in control class (to be added in next week) and in Job.*
- *Update (job status) – operation updateStatus() of Job.*
- *Record (completion of … and commencement of task) – operations complete() and start() of Task.*
- *(functionality is available …) – named associations <available> between the respective User accounts and Job. Consider also using roles for the association (this option is implemented in the diagram).*
- *Alert (display) – could be captured as a named association between Alert and Managers. However, due to using roles with the association, the association name is not shown in the class diagram.*
- *Exceed .. time period – define isCueNeeded() operation for Office/Shift Manager classes.*
- *Should be performed – ignored, a generic phrase, no further modeling consequences.*
- *To pay – operation makePayment() of CustomerAccount.*
- *Had placed, have completed – ignored, no modeling consequences.*
- *Is connected to – ignored in in analysis. In design, this statement will be revisited.*
- *To move (funds) – ignored, no modelling consequences.*
- *(payment) Is recorded – named association <is recorded> between Job and Payment.*
- *Can be paid – another reference to the same association but refers to multiplicity in the association.*
- *(payment in full) Is accepted – defines a constraint on the amount of payment, which must be derived from the job amounts (sum of task prices).*
- *Includes – ignored, a generic term, no modeling consequences.*
- *Setting up – named association <sets up> between UserAccount and OfficeManager.*
- *Must be implemented – a reference to the user roles that must be offered by BAPERS. No modeling consequences.*

*As you can appreciate the analysis is extensive. Although it seems mechanistic it requires passing a **reasoned judgment** about what the terms/phrases imply for the model. There is no simple rule to follow.*

*Clearly most of the identified nouns/verbs, especially verbs, turn out **not** to lead to useful modelling concepts. This is typical for textual descriptions of the problems and merely reiterates the point that extracting a useful analysis model from a textual description requires patience and effort. Being able to abstract from a given text is a skill which requires effort and persistence to master.*

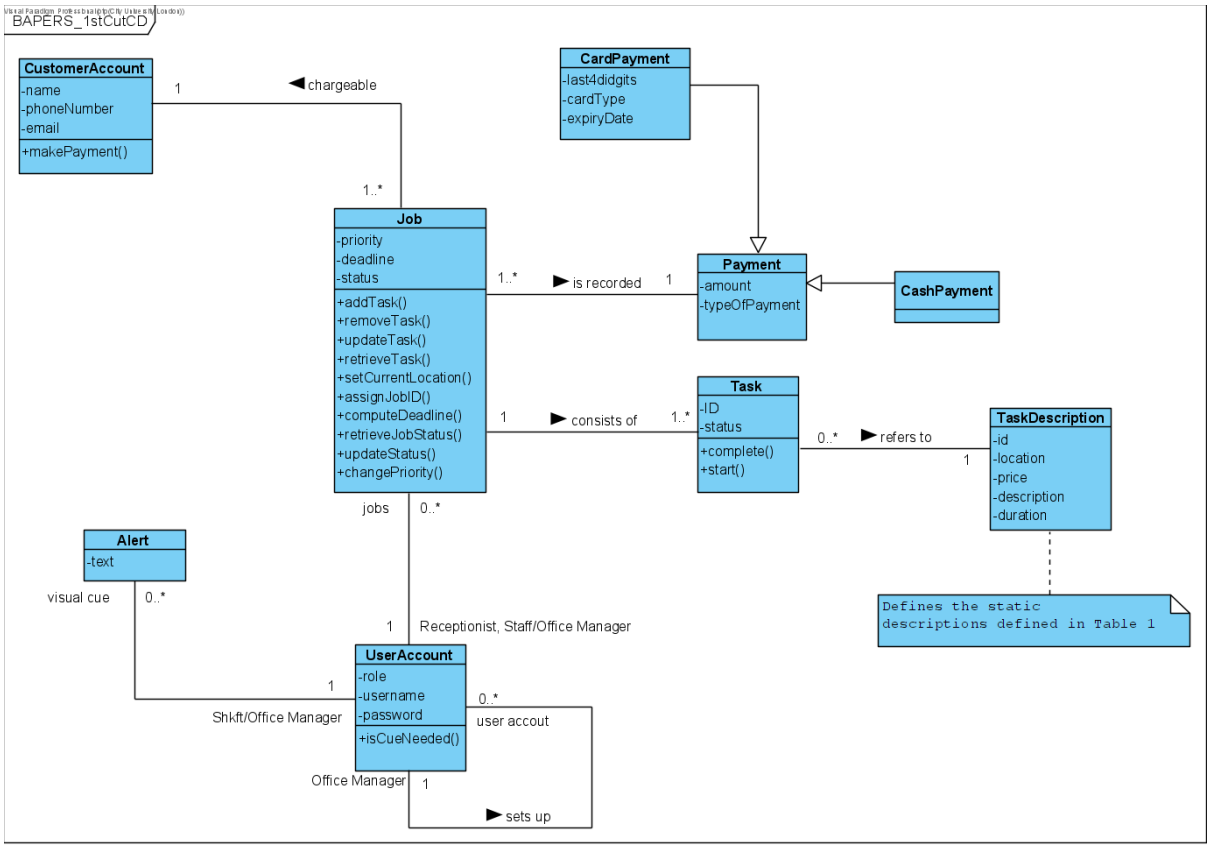
Exercise 2:

Develop an analysis (1st cut) class diagram for BAPERS showing the ***problem domain classes only*** and their associations (names/roles, multiplicities, and direction of the associations). Identify all generalisations between problem domain classes.

Model answer:

A plausible solution is provided below. The diagram is included in the VP project released with this model answer on Moodle.

*Note the use that I used **roles** with the class UserAccount, which makes the model concise. An alternative option is to introduce sub-classes to model different roles. Using inheritance is justified as different roles will have different associations with the other classes.*
Note, however, that CashPayment subclass looks obsolete, as it does not offer any difference with the superclass Payment. We will revisit this model next week.



Created: 18th September 2020

Last updated: 1st September 2025

Peter Popov

Appendix

Bloomsbury's Automated Process Execution Recording System (BAPERS)

The case study describes the requirements for automating the processes at the photographic laboratory 'The Lab' operated by Bloomsbury's Image Processing Laboratory (**BIPL)**. **BIPL** handles the work of professional photographers and carries out *jobs* on behalf of customers, but the customers do not use the system. The access to the system is limited to the laboratory staff only.

Each job is given a *unique identifier* and assigned *urgency* (a job may be urgent or normal which affects the turnaround time) when the job is placed. Urgent jobs must be completed within 6 hours while the normal jobs – within 24 hours. A job may consist of any number of the *standard tasks* offered by BIPL.

Currently there are around 30 standard tasks. Each task has an identifier (e.g., a number), is carried out at a particular location within the laboratory, has a quoted price, and an (estimated) duration. Some of the tasks are shown in Table 1 below. The case study ***does not require*** that the list of tasks be maintained (i.e. BAPERS does NOT offer functionality for adding new tasks, removing, or updating existing tasks).

Table 1

Task ID	Task Description	Location	Price (£)	Duration (min)
1.	Use of large copy camera	Copy Room	19.00	120
2.	Black and white film processing	Development Area	49.50	60
3.	Bag up	Packing Department	6.00	30
4.	Colour film processing	Development Area	80.00	90
5.	Colour Transparency processing	Development Area	110.30	180
6.	Use of small copy camera	Copy Room	8.30	75
7.	Mount Transparencies	Finishing Room	55.50	45
…	Etc.	Etc.	Etc.	Etc.

At any given time, hundreds of jobs will be in progress or pending within the laboratory. Every accepted job must be chargeable to a valid *customer account*, either an existing account, or a newly created account (e.g., a photographer may walk in, leave a film to be developed, and pay cash on collecting the finished job).

BIPL want to enable the employee on the reception desk to enter the job on a computer terminal. The material will be labelled with the job number and taken down to the laboratory.

The laboratory staff will interrogate **BAPERS** to ascertain the tasks required. As the job is transferred from one location to another in the laboratory, the staff responsible for each task will record its completion on a computer terminal in their location before passing it on. A terminal will be required in each of the following locations: Copy Room, Dark Room, Development Area, Printing Room, Finishing Room, and Packing Department.

Many jobs will be going through the laboratory at any given time, and confusion between them must be avoided. At all cost loss or mistreatment of the customer's material must not occur. Queues of work may build up at the processing stations, but flexible scheduling is required to allow priority to be given to *urgent* jobs over the *regular/normal* jobs. The system should provide functionality for inspecting the list of active/pending jobs as well as already completed ones, including the inspection of the progress of individual tasks (active/pending tasks vs. completed ones).

BAPERS must therefore provide the following main facilities:

BAP-ACCT *Accept job at reception*: Identify existing (or create new) customer account (name, phone). Assign job number. Record the deadline for completion of the job. This functionality will be mainly used by the receptionists but will also be made accessible by either Office manager or Shift manager in case receptionist(s) are absent.

BAP-PROC *Process a job through laboratory*: Respond to enquiries from any computer terminal about status of any jobs in progress, or of all jobs (including the completed ones). Update status of any given job by recording completion of current task and commencement of next (possibly with transfer of material to a new location). This functionality is available to Technicians, Shift and Office Manager.

In addition, alert Shift and/or Office manager (by, for example, displaying a visual cue with appropriate text) if the expected time to complete outstanding tasks for any job is likely to exceed the set time period, i.e., if the deadline for the job is not likely to be met; the alerts should be performed only for these two user roles.

BAP-PAYM *Payment processing*. The customers are supposed to pay once the jobs they had placed have been completed. Customers can pay by cash or credit/debit card only. The system is connected to an external payment processor to move funds between the customers' credit/debit cards and the BIPL bank account. In case of a card payment BAPERS is expected to connect to the external payment processor and on successful completion of the card payment the payment is recorded with the following details: card payment, type of card, and the last 4 digits of the card used. Cash payments are recorded, too. Several jobs can be paid at once. Only payment in full is accepted (i.e., no partial payments are allowed). This functionality is available to Receptionist, Shift and Office Manager.

BAP-ADMN *Administering the system.* This includes creating a user account for BAPERS and setting up access privileges. The following user roles are essential to be implemented: Office manager, Shift manager, Receptionist and Technician. This functionality will be available to Office Manager only.

Created: 20th September 2019

Last updated: 12th September 2023

Peter Popov