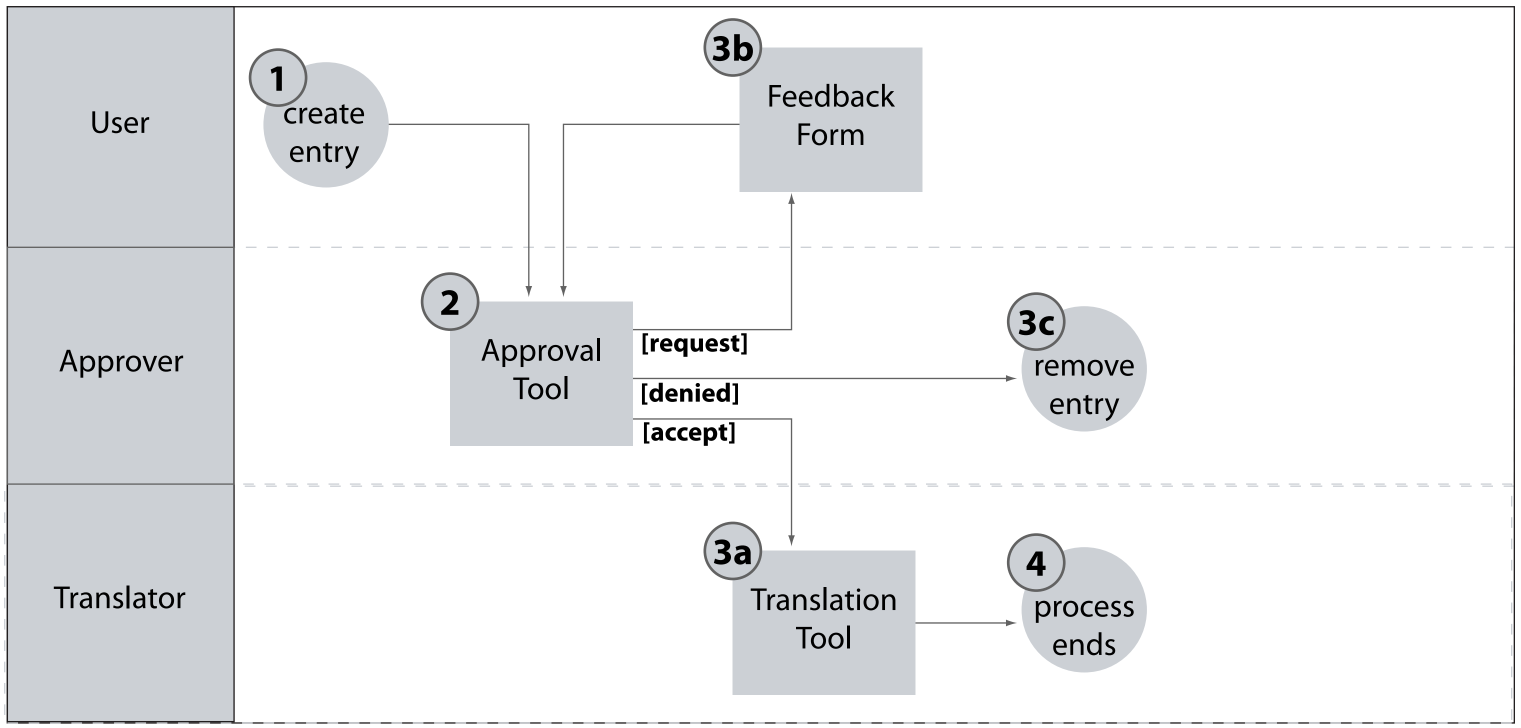
Approval Tool

We want to design a generic interface for an administrator to be able to approve different transactions in the system. Each approval can either be [accept], [denied] or [request].

Transaction Process:



1. user creates a new entry in the system
2. an administrator reviews the entry and either: [accepts], [rejects] or [requests info]

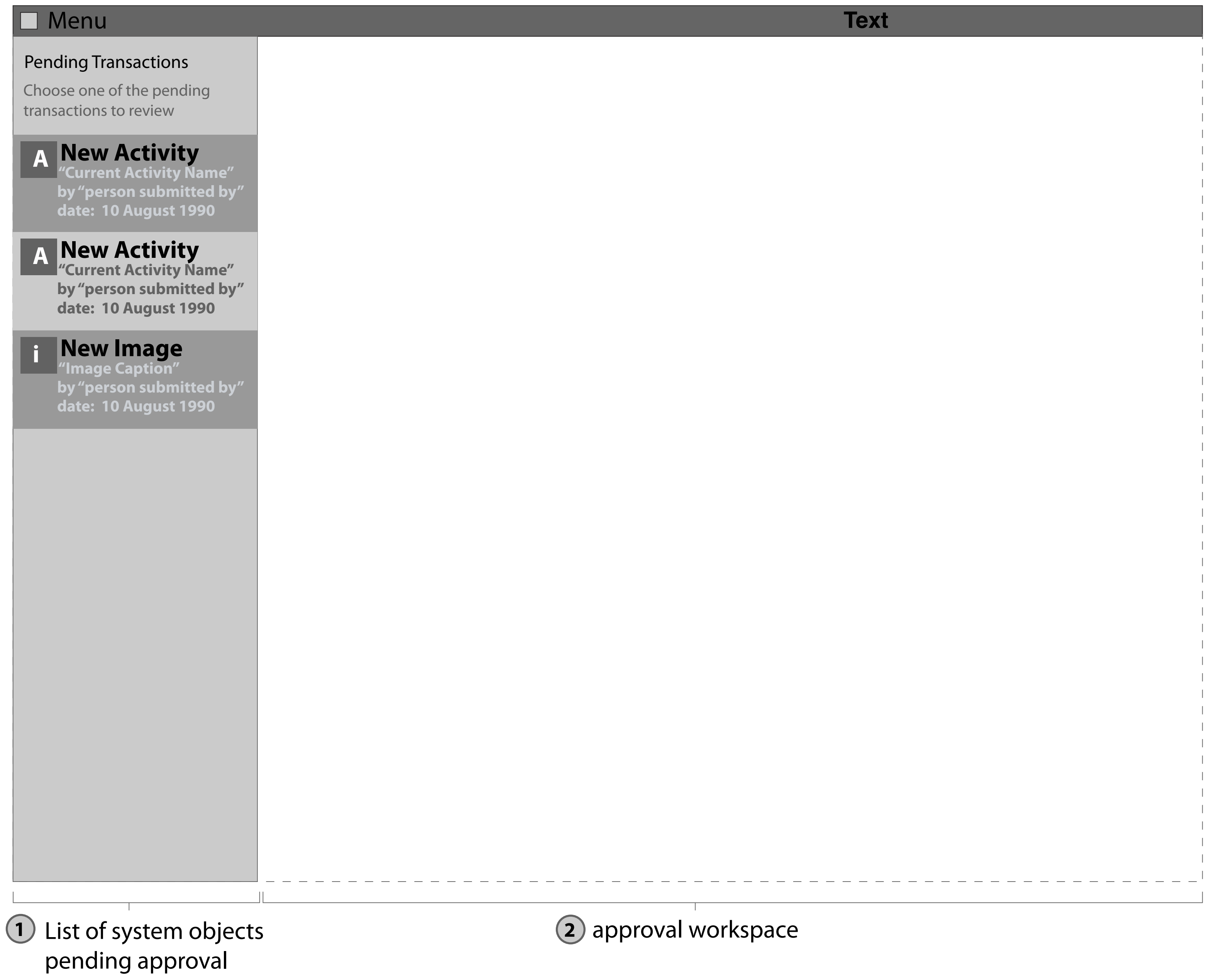
3a [accept]: entry can move on to the next step in it's process. (translation)

3b [request info]: approval entry is now shown to user

3c [denied]: entry is removed and process is stopped.

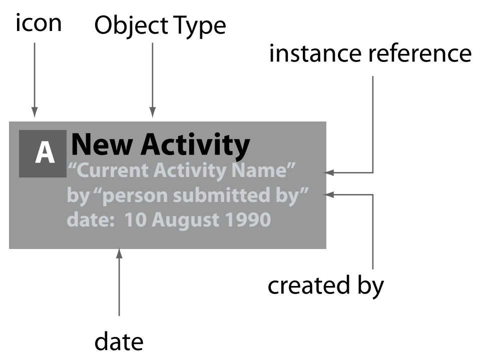
Approval Tool - layout

The approval Tool will have the following layout



1. A scrollable list of all approvals (status=="pending")

   An individual entry in the list will have the following layout:

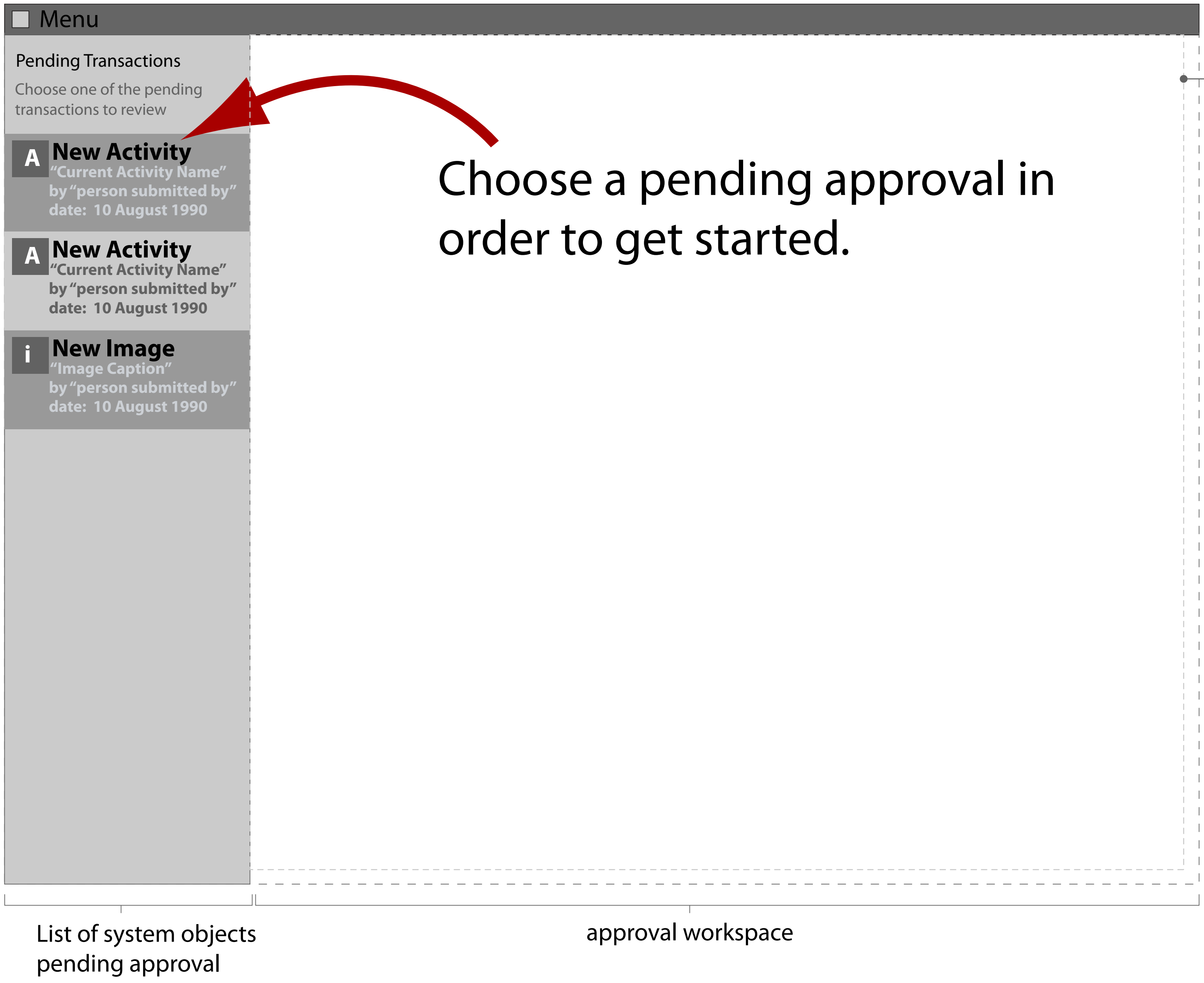


(The icon should be a .class reference)

2. The work area for any approval forms that will be displayed.

Approval Tool - layout - instructions

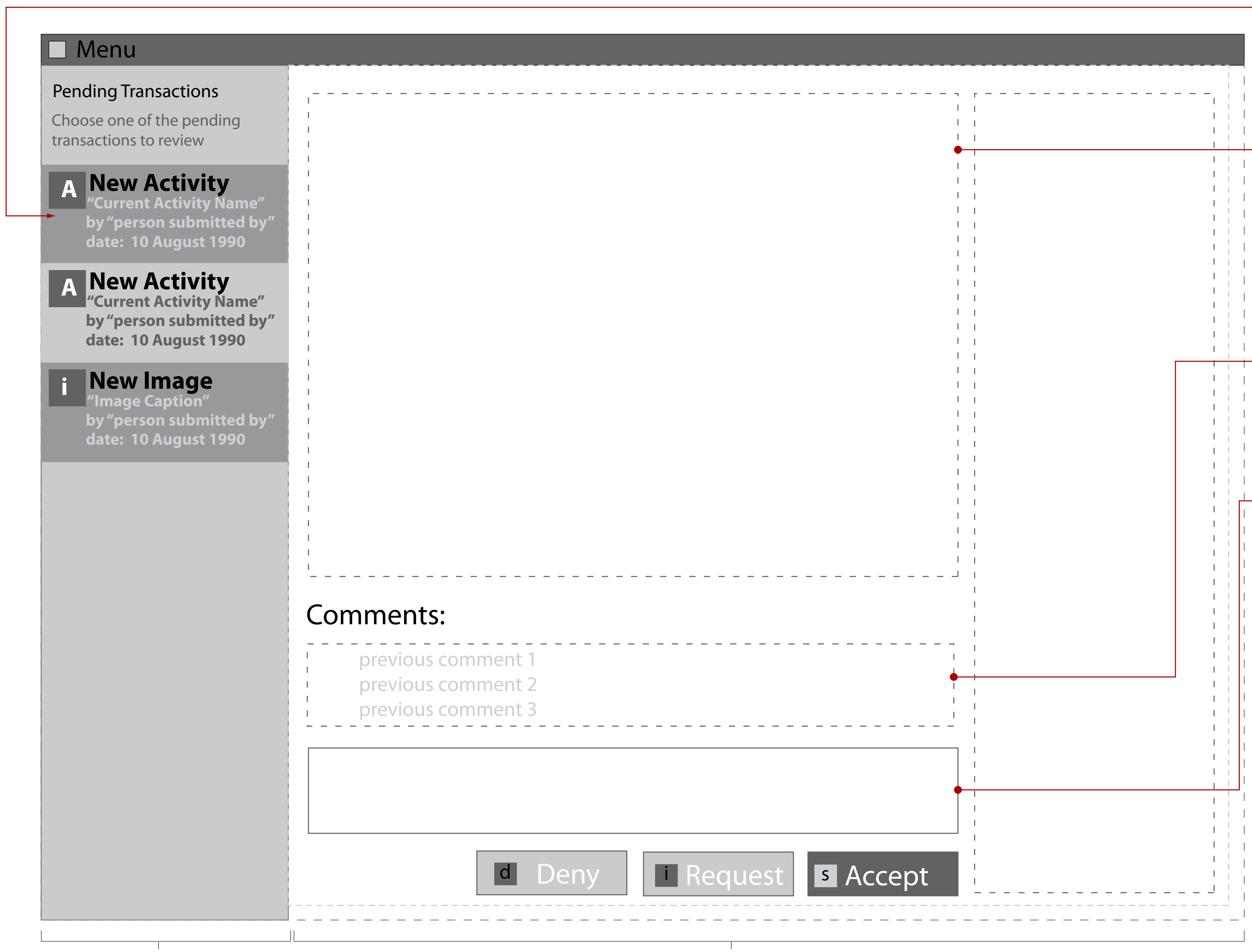
When the tool first loads, the work area will have a set of instructions



(1) The InstructionsPanel simply displays a set of instructions to tell the administrator how to get started.

Approval Tool - layout - Approval Form

When the user clicks on an entry in the list, the workspace is replaced with the Approval Form


Menu
Pending Transactions
Choose one of the pending transactions to review
A New Activity "Current Activity Name" by "person submitted by" date: 10 August 1990
A New Activity "Current Activity Name" by "person submitted by" date: 10 August 1990
i New Image "Image Caption" by "person submitted by" date: 10 August 1990
Comments:
previous comment 1
previous comment 2
previous comment 3
d Deny
i Request
s Accept

List of system objects pending approval

approval workspace

1. When the user clicks on the List
2. We display the Approval form

Unique Object Display Form

This area displays a Form unique to the object being approved.

Comment Area

if an object has been [request] one or more times, then the previous comments/questions/responses will be displayed here.

There is a text box for the administrator to enter any comments for this approval.

A comment is required for the [request] action.

Action Buttons

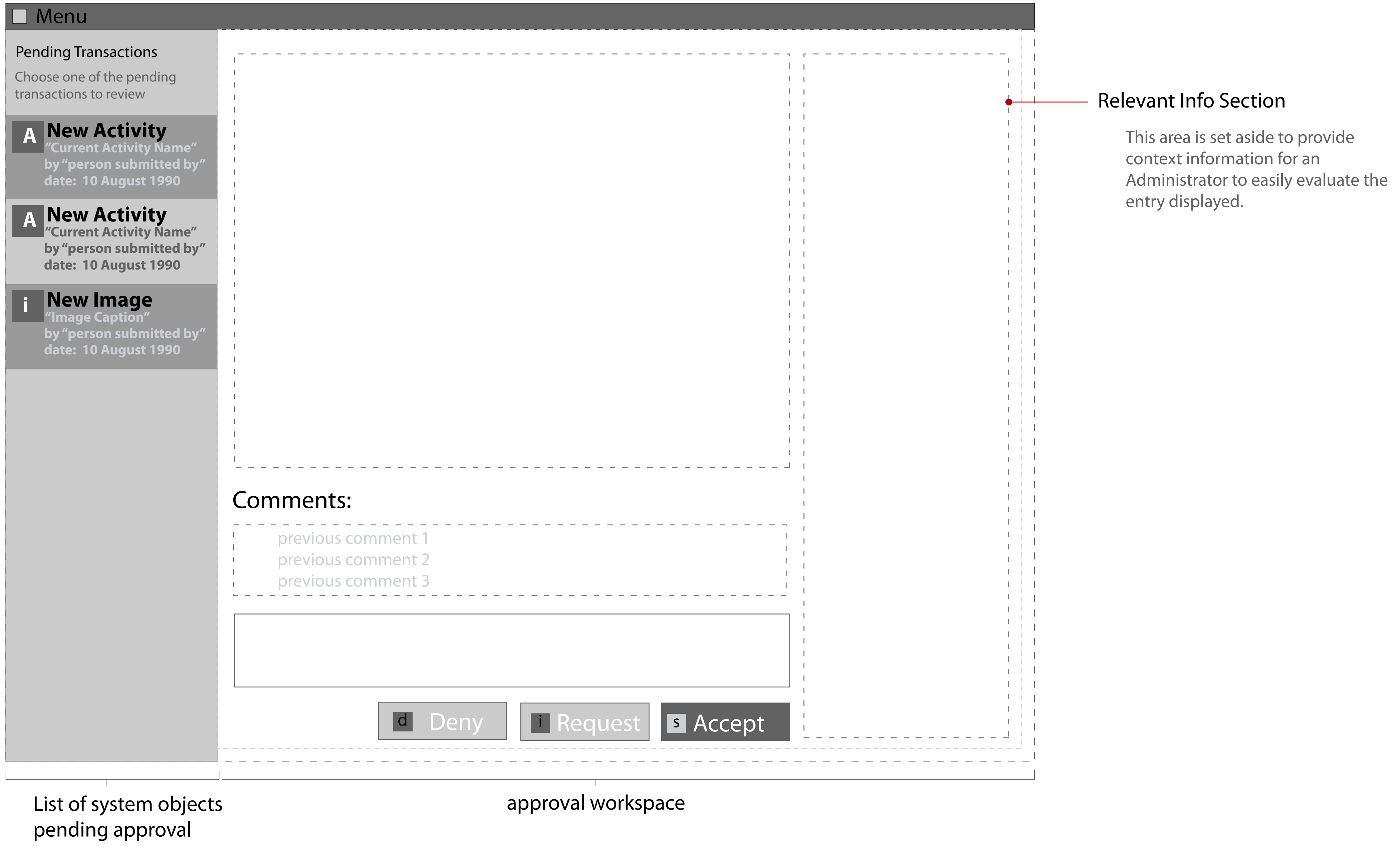
[accept] updates the object with any changes to it's current definition in the object display form. And changes the status=="accepted"

[deny] Changes the status=="denied"

[request info] Changes the status=="request"

Approval Tool - layout - Approval Form

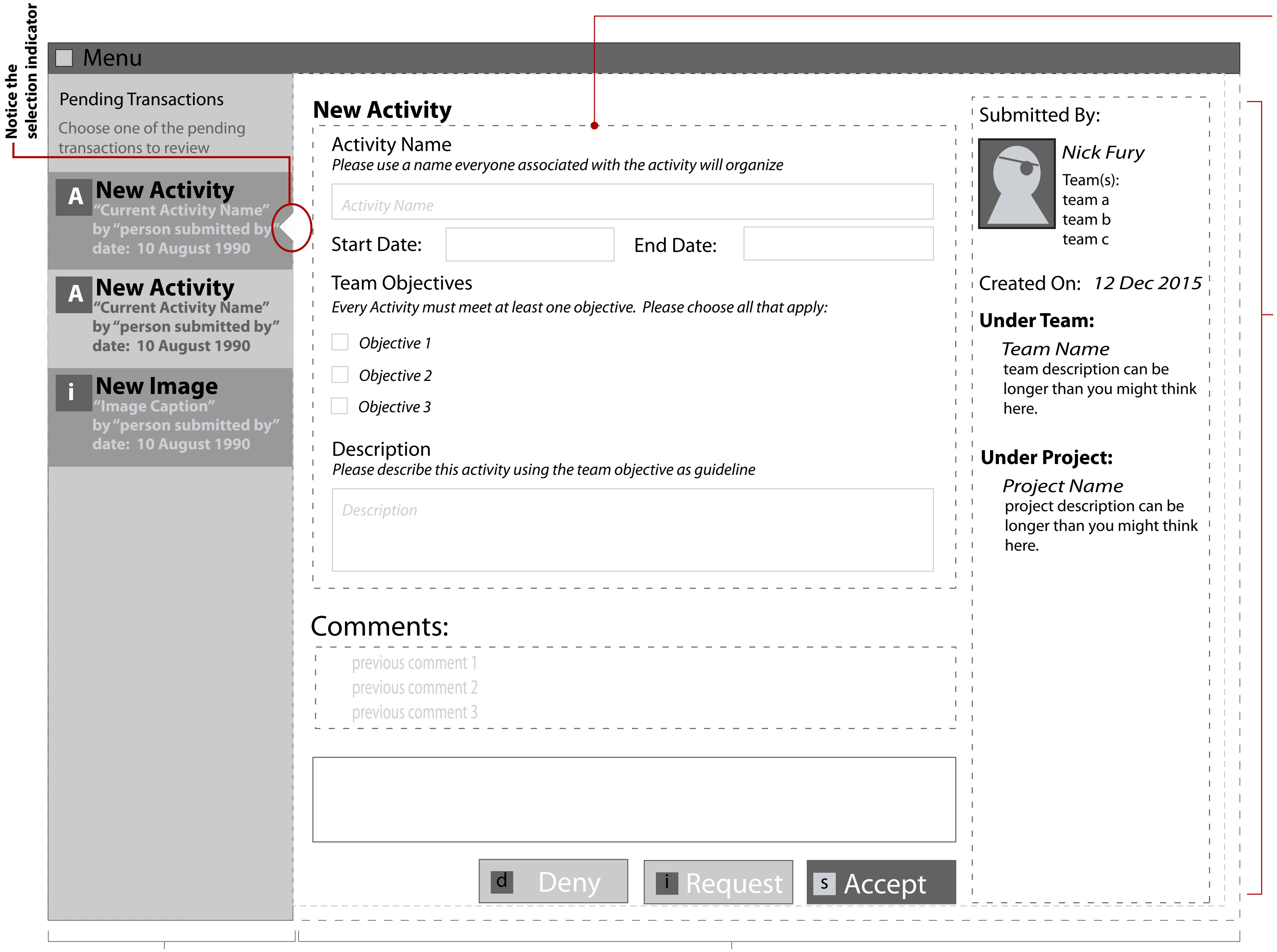
When the user clicks on an entry in the list, the workspace is replaced with the Approval Form



This area is set aside to provide context information for an Administrator to easily evaluate the entry displayed.

Approval Tool - layout - Approval Form - Activities

The Unique Object Display for the Activity Report. We display an Activity Object Form, and information relavant to that Activity.



Pending Transactions

Choose one of the pending transactions to review

A **New Activity**
"Current Activity Name"
by "person submitted by"
date: 10 August 1990

A **New Activity**
"Current Activity Name"
by "person submitted by"
date: 10 August 1990

i **New Image**
"Image Caption"
by "person submitted by"
date: 10 August 1990

New Activity

Activity Name
Please use a name everyone associated with the activity will organize

Activity Name

Start Date: End Date:

Team Objectives
Every Activity must meet at least one objective. Please choose all that apply:

- ☐ *Objective 1*
- ☐ *Objective 2*
- ☐ *Objective 3*

Description
Please describe this activity using the team objective as guideline

Description

Comments:

previous comment 1
previous comment 2
previous comment 3

Submitted By:

Nick Fury
Team(s):
team a
team b
team c

Created On: *12 Dec 2015*

Under Team:

Team Name
team description can be longer than you might think here.

Under Project:

Project Name
project description can be longer than you might think here.

1 The Object Data Form

This section redisplays the object data in a form that the Administrator can view / edit.

2 A column showing relevant info for the data.

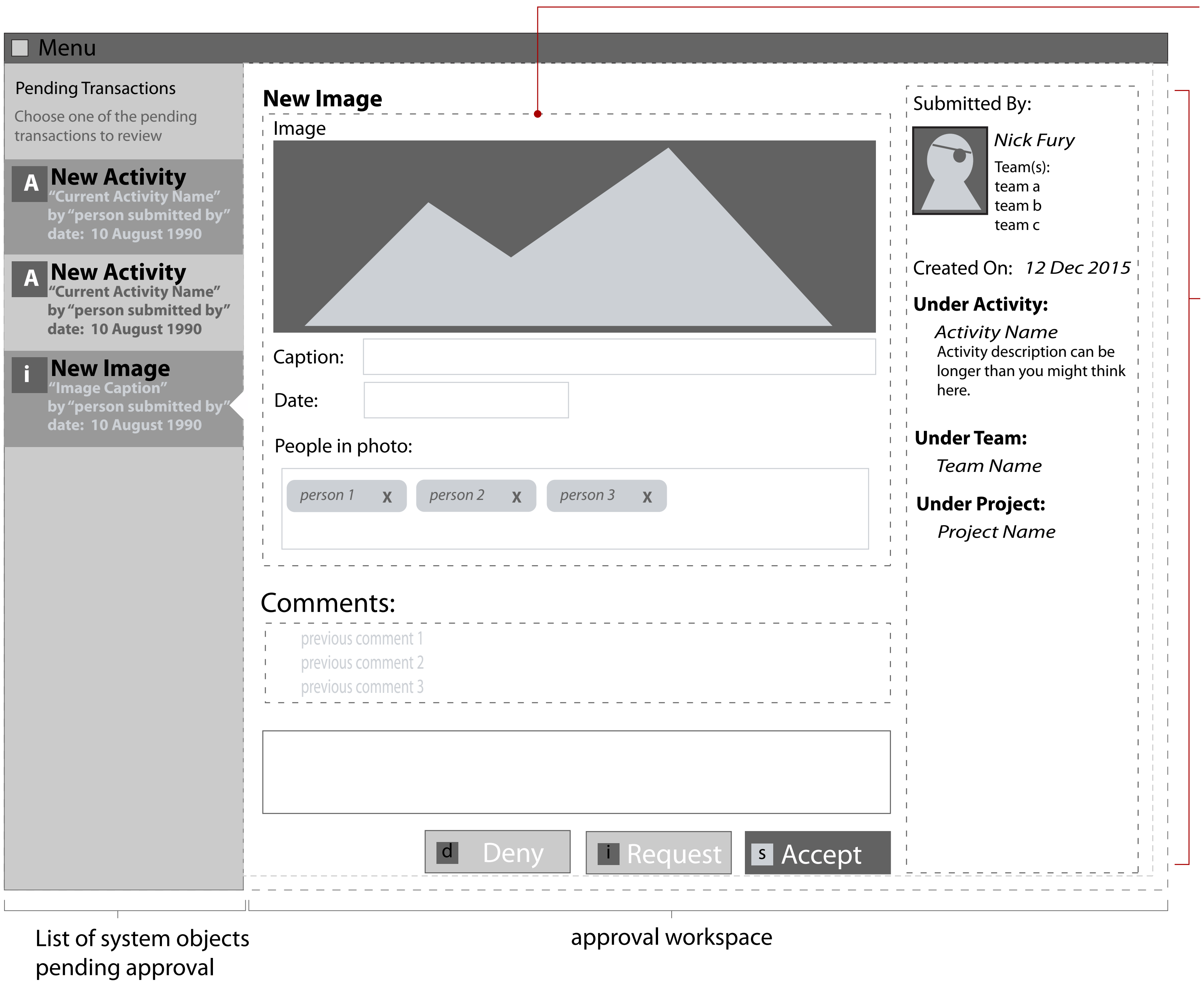
This section displays relevant information associated with the object to help the Administrator decide if this entry should be [accept] or [deny]

List of system objects pending approval

approval workspace

Approval Tool - layout - Approval Form - Images

The Unique Object Display for the Image Approval form:



Menu

Pending Transactions

Choose one of the pending transactions to review

A **New Activity**
"Current Activity Name"
by "person submitted by"
date: 10 August 1990

A **New Activity**
"Current Activity Name"
by "person submitted by"
date: 10 August 1990

i **New Image**
"Image Caption"
by "person submitted by"
date: 10 August 1990

New Image

Image

Caption:

Date:

People in photo:

person 1 x *person 2* x *person 3* x

Comments:

previous comment 1
previous comment 2
previous comment 3

d Deny
i Request
s Accept

Submitted By:

Nick Fury
Team(s):
team a
team b
team c

Created On: *12 Dec 2015*

Under Activity:
Activity Name
Activity description can be longer than you might think here.

Under Team:
Team Name

Under Project:
Project Name

List of system objects pending approval

approval workspace

1 The Object Data Form

This section redisplays the object data in a form that the Administrator can view / edit.

2 A column showing relevant info for the data.

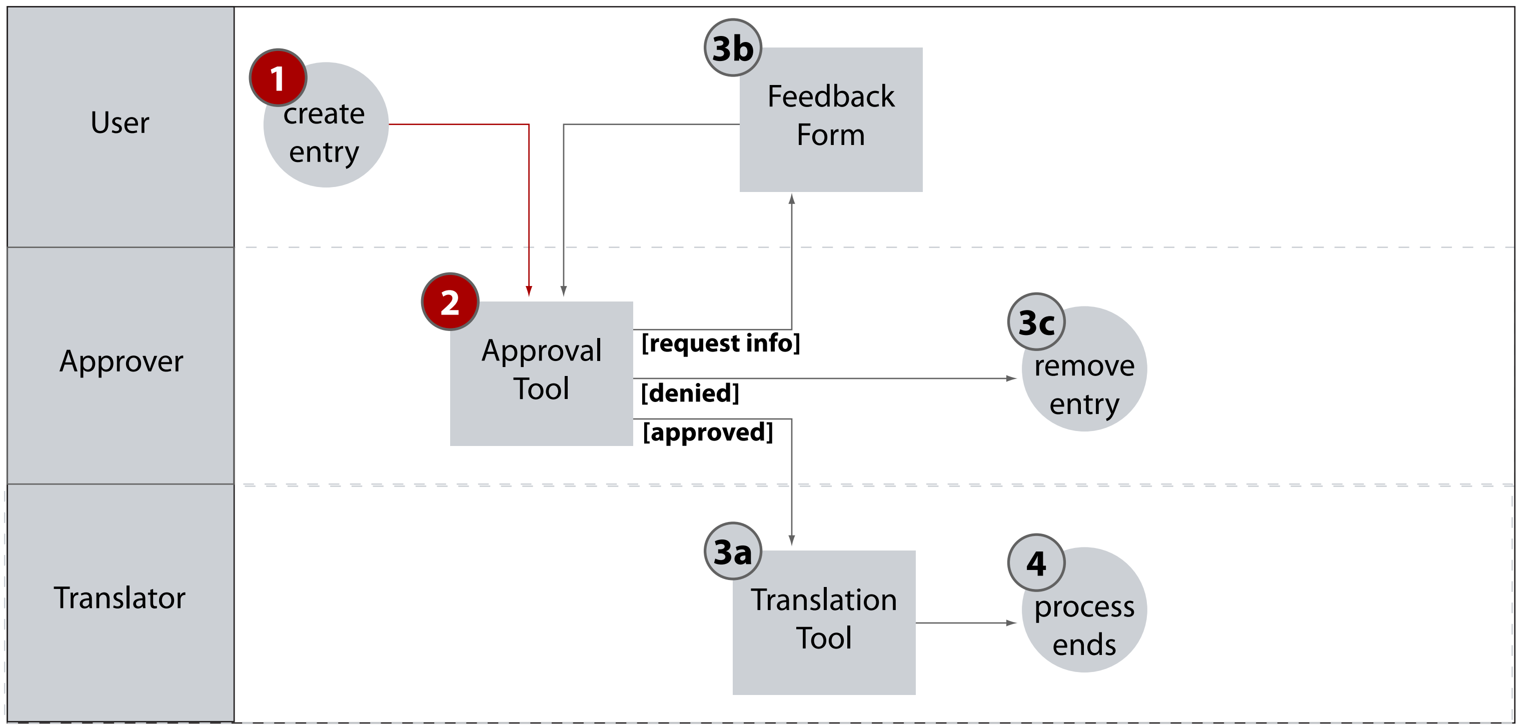
This section displays relevant information associated with the object to help the Administrator decide if this entry should be [accept] or [deny]

Approval Tool - Implementation - New Approval

Stepping through the process of how this tool will work.

To begin with, another tool in the system will need to register an object that needs approval:

Transaction Process:



1 user creates a new entry in the system

2 an administrator reviews the entry and either: [accepts], [rejects] or [requests info]

3a [accept]: entry can move on to the next step in it's process. (translation)

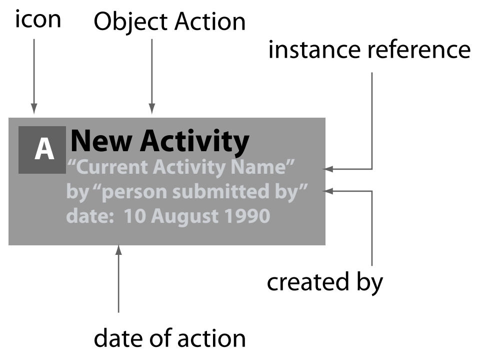
3b [request info]: approval entry is now shown to user

3c [denied]: entry is removed and process is stopped.

Approval Tool - Implementation - New Approval

In order for the approval tool to properly display an object requiring approval, we will need to know the following information:

1 What to display for the menu entry:



For the approval tool, we want to offer a consistent look and feel for what shows up in our lists.

To create this entry we need the following json data:

```
{
  icon:'.class',
  action: { label reference },
  instanceRef:'field',
  createdBy:'user.guid',
  date:'yyyy/mm/dd'
}
```

class definitions:
Our calling tool should define a ui css class definition that defines the icon for this entry.

Multilingual Labels:
Our system is designed to support multiple languages at the same time. So in order to know which language version to display, we simply provide a multilingual label reference:

```
{
  key:'labelKey',
  context: 'context'
}
```

instanceRef:
The data displayed as the instance reference is going to be taken from the data provided to be approved. This is asking which field of that data to use for the instance ref.

```
{
  "field":'Activity Name',
  "field2":'Fun stuff'
}
```

user.guid:
The Global User ID of the user will be used to return their display name. We'll look that up at run time.

Resulting approval data:

```
{
  menu: {
    icon:'.class',
    action: {
      key:'labelKey',
      context:'context'
    },
    instanceRef:'field',
    createdBy:'user.guid',
    date:'yyyy/mm/dd'
  }
}
```

Approval Tool - Implementation - New Approval

In order for the approval tool to properly display an object requiring approval, we will need to know the following information:

2 What to display for the Form entry:

Our Approval tool isn't supposed to know anything about the external tools UI. So the tool will need to provide the UI with:

data: the data to display
view: a url to load the view to properly display the data.
viewData: *(optional)* additional data to send to the view

```
{
  data:{ definition },
  view:'uri string',
  viewData:{ definition }
}
```

data definition:
The data to be displayed will be defined in a json object as a 'field':'value' hash:

```
{
  activity_name:'New Activity',
  language_code:'en',
  start_date:'yyyy/mm/dd',
  end_date:'yyyy/mm/dd',
  activity_description: 'lots of fun.',
  objectives:[ id1, id2,.., idN]
}
```

view:
A uri string that will return the view to display the data given. It should be specified in reference to the sails /assets directory.

```
{
  view:'activities/views/activity_approval.ejs'
}
```

viewData:
Additional data required for the View to display properly. For example, in our definition, we only sent back the id's for the objectives, so we need to do a lookup for the actual Objective info:

```
{
  viewData:{
    objectives:{
      model: 'ui.model.ref',
      filter:{ team: XX }
    }
  }
}
```

on the client, this will do:
Model.find({ team:xx}) and the results will be sent to the view as: { objectives: [results] }

Resulting approval data:

```
{
  menu: {
    icon:'.class',
    action: {
      key:'labelKey',
      context:'context'
    },
    instanceRef:'field',
    createdBy:'user.guid',
    date:'yyyy/mm/dd'
  },
  form: {
    data:{ definition },
    view:"uri string",
    viewData: { definition }
  }

}
```

Approval Tool - Implementation - New Approval

In order for the approval tool to properly display an object requiring approval, we will need to know the following information:

3 What to display for the Related Info:

Like the Form entry, the related info is only going to be known to the external tool.   So we will need to provide:

view: a url to load the view to properly display the data.
viewData: data to send to the view

```
{
  view:'uri string',
  viewData:{ definition }
}
```

view:
A uri string that will return the view to display the data given. It should be specified in reference to the sails /assets directory.

```
{
  view:'activities/views/activity_approval.ejs'
}
```

viewData:
The data required for the View to display properly. For example, in our definition, we only sent back the id's for the objectives, so we need to do a lookup for the actual Objective info:

```
{
  viewData:{
    user:{
      data: {
        displayName:'Nick Fury',
        avatar:'uri/to/image.jpg',
        teams:[ id1, id2, ... idN]
      }
    },
    teams:{
      model:'ui.model.reference'
    },
    selectedTeam:id,
    projects:{
      model:'ui.model.reference'
    },
    selectedProject:id
  }
}
```

Here the data for user is sent specifically to the view.   teams, and projects are looked up by the client side models, and the team.id and project.id will be used by the view to pull out the proper entries to display.

Resulting approval data:

```
{
  menu: {
    icon:'.class',
    action: {
      key:'labelKey',
      context:'context'
    },
    instanceRef:'field',
    createdBy:'user.guid',
    date:'yyyy/mm/dd'
  },
  form: {
    data:{ definition },
    view:"uri string",
    viewData: { definition }
  },
  relatedInfo:{
    view:'uri string',
    viewData:{ definition }
  }

}
```

Approval Tool - Implementation - New Approval

In order for the approval tool to properly display an object requiring approval, we will need to know the following information:

4 What to know about the calling tool

Once the Approval tool receives an update from the user, it needs to notify the calling tool with the results.

message: the pub/sub message queue to use to return the updated info

reference: *(optional)* any json data helpful for the calling tool to process this transaction.

```
{
  message:'queue name'
}
```

message:
We'll use a pub/sub message queue to communicate back to the calling tool. The

```
ADCore.comm.publish(message, {
  status:'approve',  // or 'deny', 'request'
  data: { the resulting data object },
  reference: {}      // if provided
});
```

When the message is published, the subscription tool will send the status value of the approval, and the latest updated data that was returned from the UI.

Resulting approval data:

```
{
  menu: {
    icon:'.class',
    action: {
      key:'labelKey',
      context:'context'
    },
    instanceRef:'field',
    createdBy:'user.guid',
    date:'yyyy/mm/dd'
  },
  form: {
    data:{ definition },
    view:"uri string",
    viewData: { definition }
  },
  relatedInfo:{
    view:'uri string',
    viewData:{ definition }
  },
  callback:{
    message:'message queue'
  }

}
```

Approval Tool - Implementation - New Approval

In order for the approval tool to properly display an object requiring approval, we will need to know the following information:

5 How to know which administrator can approve this?

There might be several users assigned to Admin roles, in our portal. So we need a way to make sure the requesting user has permission to process this approval.

In our framework, this is accomplished using:

actionKey: the actionKey a user needs to have access to in order to process this entry.
userid: the user.guid of the user who created the entry.

```
{
  actionKey:'tool.action.verb',
  userid: guid
}
```

actionKey:
Different tools define unique 'action keys' in the system that determine a user's access to certain features. The calling tool will need to specify what action key is required for a given administrator to have permission to approve this transaction.

userID:
Permissions in the system are associated with a given 'scope'. This determines which users a person can use their assigned actionKey on. The provided userID will be checked against the administrators scope for the provided actionKey.

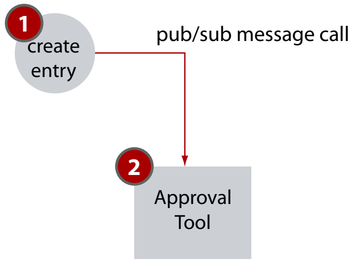
Resulting approval data:

```
{
  menu: {
    icon:'.class',
    action: {
      key:'labelKey',
      context:'context'
    },
    instanceRef:'field',
    createdBy:'user.guid',
    date:'yyyy/mm/dd'
  },
  form: {
    data:{ definition },
    view:"uri string",
    viewData: { definition }
  },
  relatedInfo:{
    view:'uri string',
    viewData:{ definition }
  },
  callback:{
    message:'message queue'
  },
  permission:{
    actionKey:'tool.action.verb',
    userID: guid
  }
}
```

Approval Tool - Implementation - New Approval

In order for the approval tool to properly display an object requiring approval, we will need to know the following information:

6 How does an external tool make the request for an object to be approved?



Approval Tool:
When the sails system boots up, the Approval tool will subscribe to a message Queue under : **opsportal.approval.create**

External Tool Subscribes:
At the point in the process when an external tool makes a change that needs approval (like creating a new resource, or modifications to existing data, etc...) they will then need to register the approval by publishing to this message Queue.

```
var transactionApproval = {
  menu: {
    icon:'.class',
    action: {
      key:'labelKey',
      context:'context'
    },
    instanceRef:'field',
    createdBy:'user.guid',
    date:'yyyy/mm/dd'
  },
  form: {
    data:{ definition },
    view:"uri string",
    viewData: { definition }
  },
  relatedInfo:{
    view:'uri string',
    viewData:{ definition }
  },
  callback:{
    message:'your.tool.queue.name',
    reference: { id: xx }
  },
  permission:{
    actionKey:'tool.action.verb',
    userID: guid
  }
}

ADCore.comm.publish('opsportal.approval.create',
transactionApproval);
```

External Tool Responds:
In order for an external tool to respond to an approval, then it also needs to subscribe to a Message Queue, and respond to the data returned by the Approval Tool.

```
ADCore.comm.subscribe('your.tool.queue.name',
function(message){

  // find original entry
  Model.find({id: message.reference.id})
  .catch(function(err) {
    // handle error
  })
  .then(function(model) {

    // save the status of this entry:
    model.status = message.status;
    model.save()
    .catch(function(err) {
      // handle error
    })
    .then(function(model) {

      // save any updated info from the approver:
      if (message.status == "approve") {

        Multilingual.model.update({
          model: model,
          data:message.data
        })
        .fail(function(err) {
          // handle error
        })
        .then(function(result) {

          // all done

        });
      }
    });
  });
});
```

Approval Tool - Implementation - Data Storage

This is how we will store the data for our approval requests.

Our approval request will look like:

```
{
  menu: {
    icon:'.class',
    action: {
      key:'labelKey',
      context:'context'
    },
    instanceRef:'field',
    createdBy:'user.guid',
    date:'yyyy/mm/dd'
  },
  form: {
    data:{ definition },
    view:"uri string",
    viewData: { definition }
  },
  relatedInfo:{
    view:'uri string',
    viewData:{ definition }
  },
  callback:{
    message:'message queue'
  },
  permission:{
    actionKey:'tool.action.verb',
    userID: guid
  }
}
```

These are the important parts of the data that our action tool will need to reference on the server:

callback:
We will need to know the message queue to post the results to.

actionKey & userID:
We need these two pieces of information to make sure we know who to send these approval requests to.

In addition to these, we'll need to keep track of:

status:
The status of the current request:
['approved', 'denied', 'requesting', 'pending']

objectData:
The remaining data that needs to be sent to the client so it can be displayed properly.

There is also going to be 1 or more comments related to an approval request. These comments need to have:

comment:
The entered comment requested by the Approver.

response:
The response submitted by the creator.

Resulting MySQL table data:

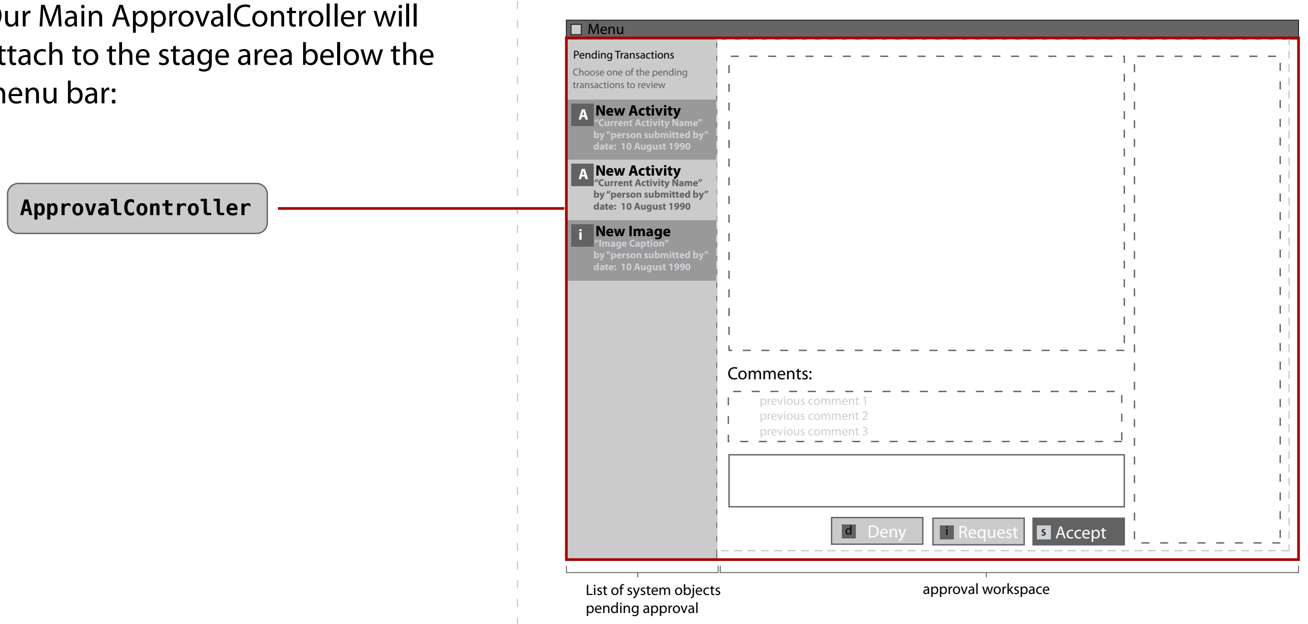
```
ApprovalRequests {
  actionKey: 'string',
  userID: 'string',
  callback:'string',
  status:'string',
  objectData:'string',
  comments:[hasMany ApprovalComments]
}
```

```
ApprovalComments {
  comment: 'string',
  response: 'string',
  request:[hasOne ApprovalRequest]
}
```

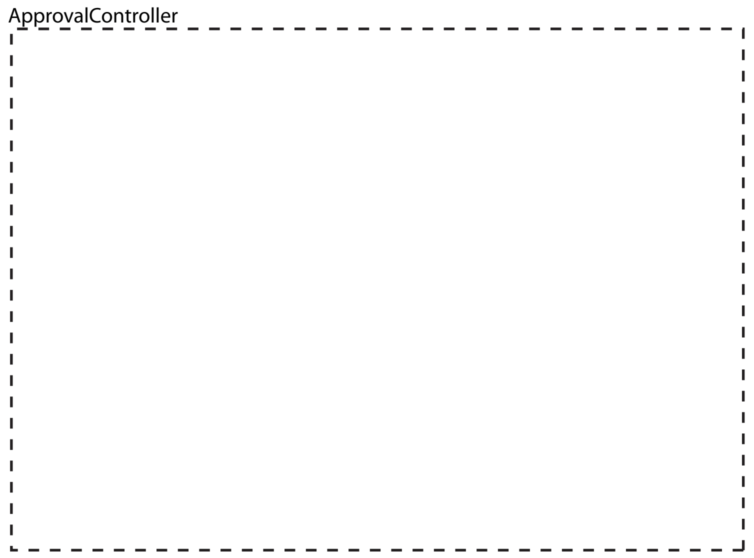
Approval Tool - Implementation - UI Controllers

Organization of the UI

Our Main ApprovalController will attach to the stage area below the menu bar:



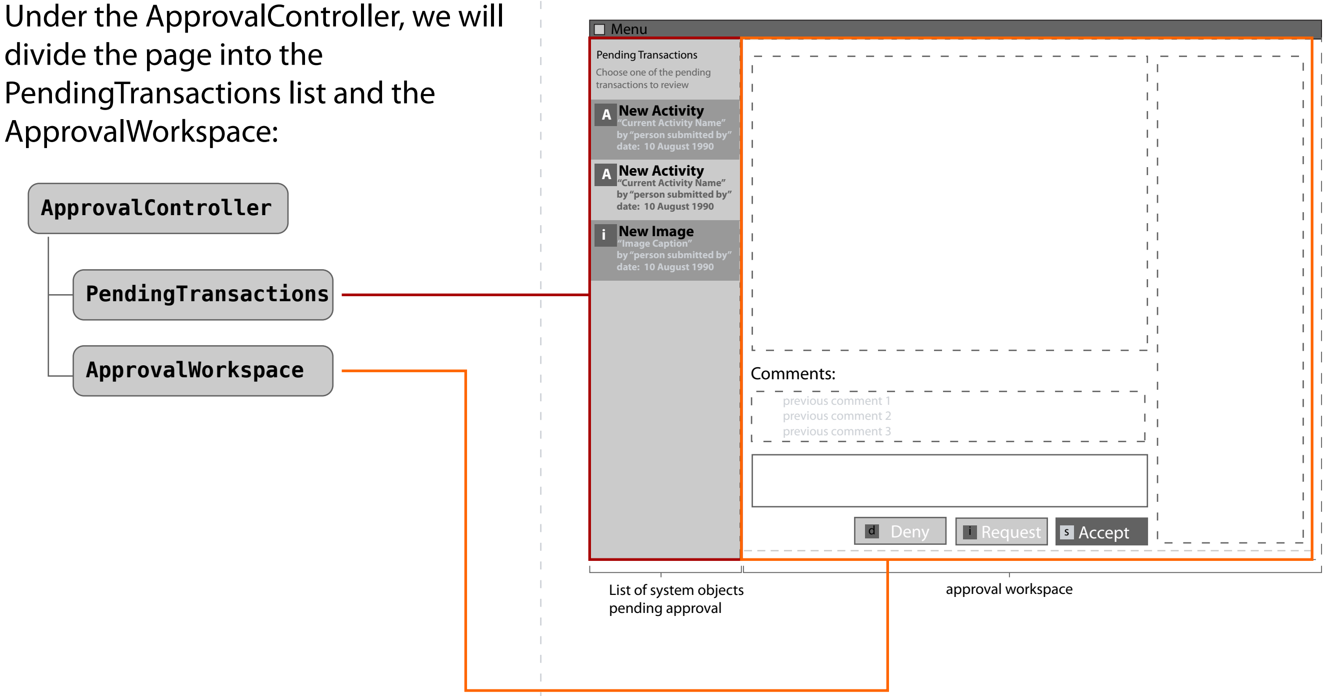
The resulting layout:



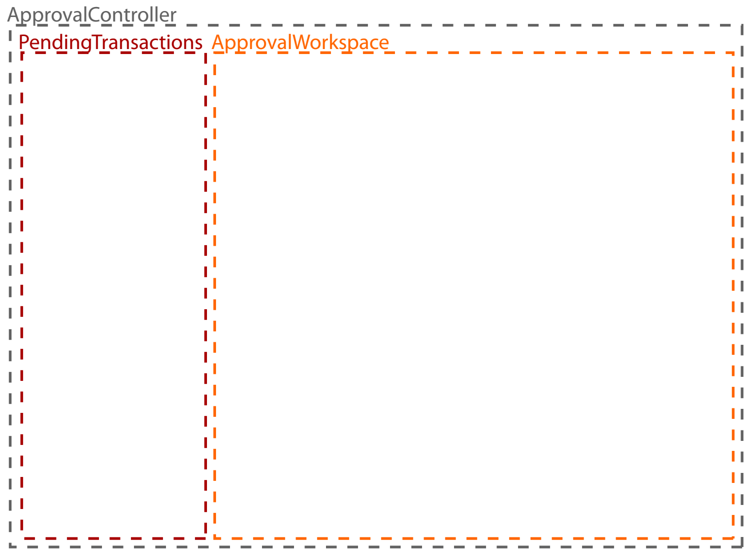
Approval Tool - Implementation - UI Controllers

Organization of the UI

Under the ApprovalController, we will divide the page into the PendingTransactions list and the ApprovalWorkspace:



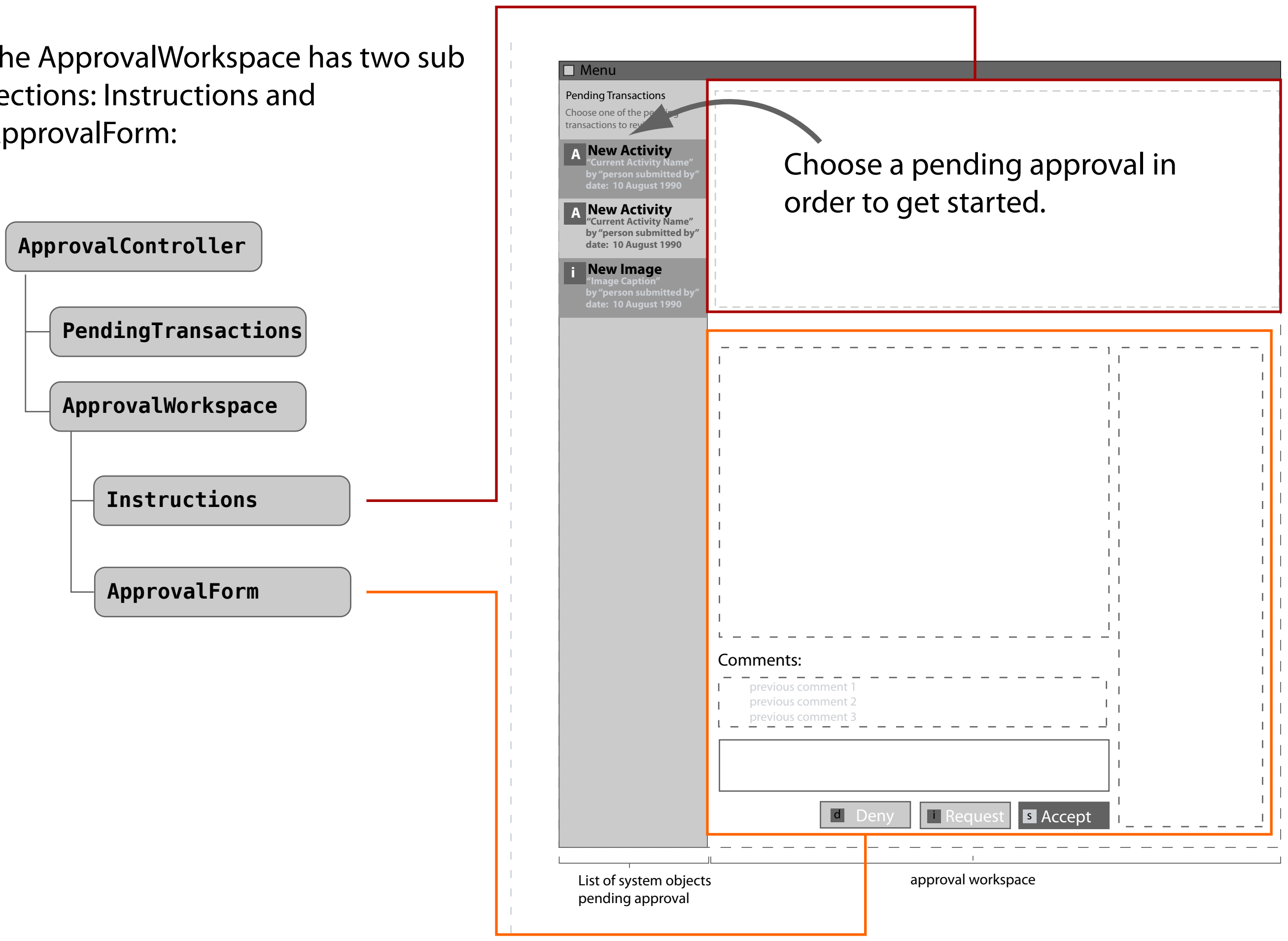
The resulting layout:



Approval Tool - Implementation - UI Controllers

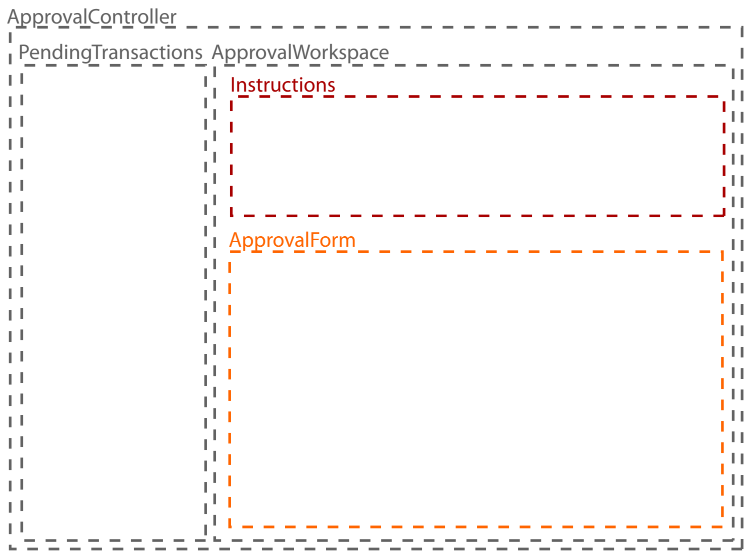
Organization of the UI

The ApprovalWorkspace has two sub sections: Instructions and ApprovalForm:



ApprovalController
PendingTransactions
ApprovalWorkspace
Instructions
ApprovalForm
Menu
Pending Transactions
Choose one of the pending transactions to review
A New Activity "Current Activity Name" by "person submitted by" date: 10 August 1990
A New Activity "Current Activity Name" by "person submitted by" date: 10 August 1990
i New Image "Image Caption" by "person submitted by" date: 10 August 1990
Choose a pending approval in order to get started.
Comments:
previous comment 1
previous comment 2
previous comment 3
d Deny
i Request
s Accept
List of system objects pending approval
approval workspace

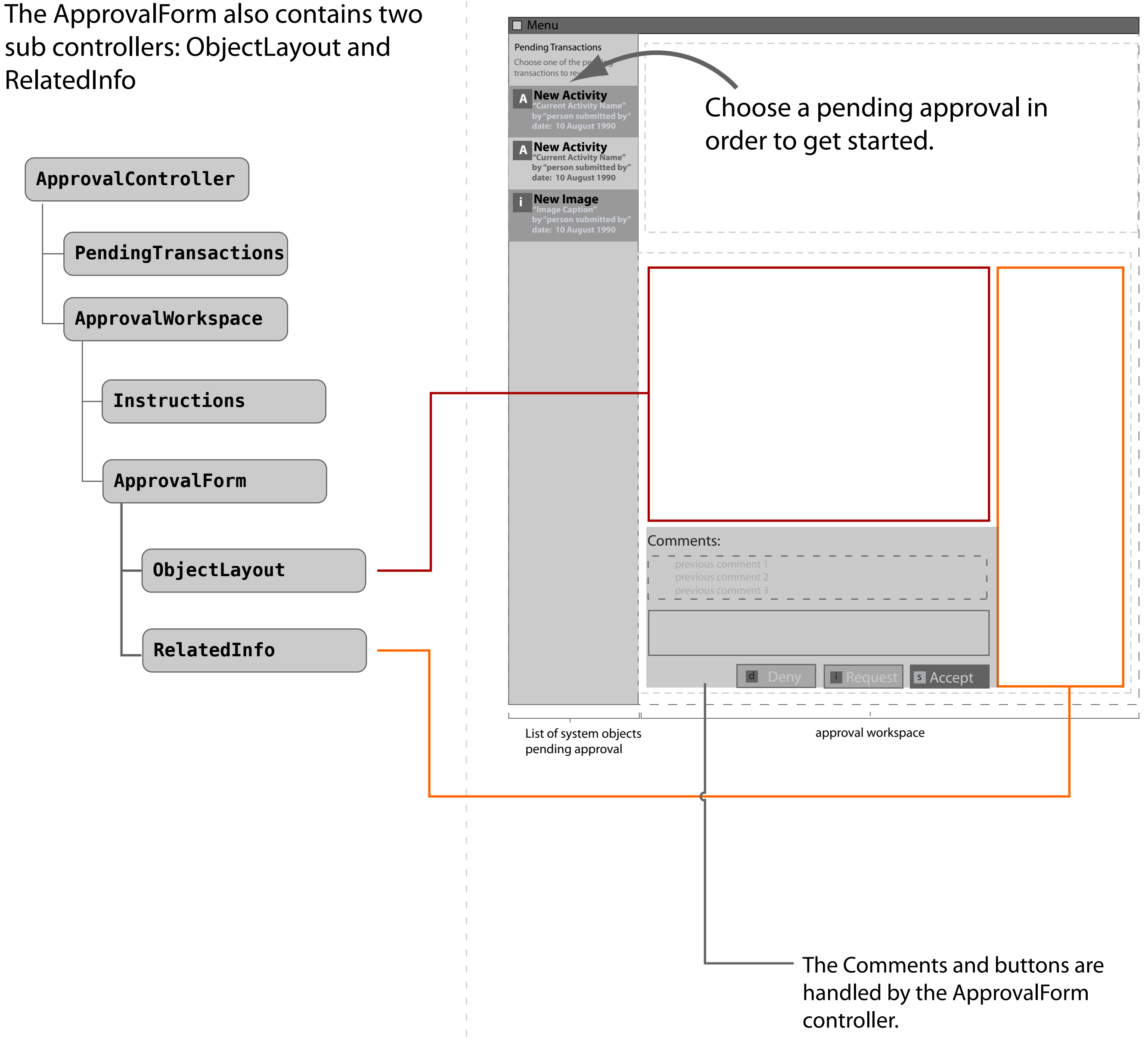
The resulting layout:



Approval Tool - Implementation - UI Controllers

When the user clicks on an entry in the list, the workspace is replaced with the Approval Form

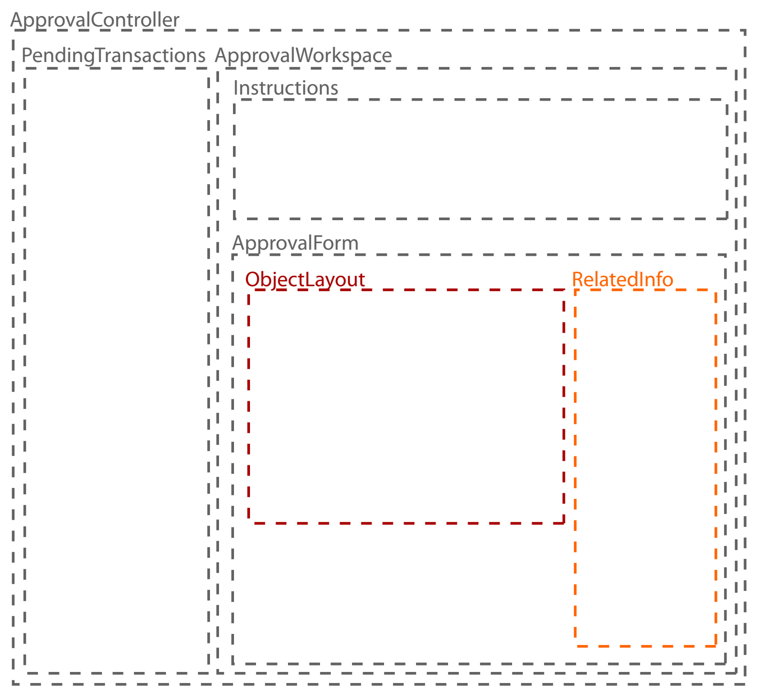
The ApprovalForm also contains two sub controllers: ObjectLayout and RelatedInfo



Choose a pending approval in order to get started.

The Comments and buttons are handled by the ApprovalForm controller.

The resulting layout:


ApprovalController
PendingTransactions
ApprovalWorkspace
Instructions
ApprovalForm
ObjectLayout
RelatedInfo