# SSN SASE ONLINE APPLICATION SYSTEM

## Part 1 - Business Context :

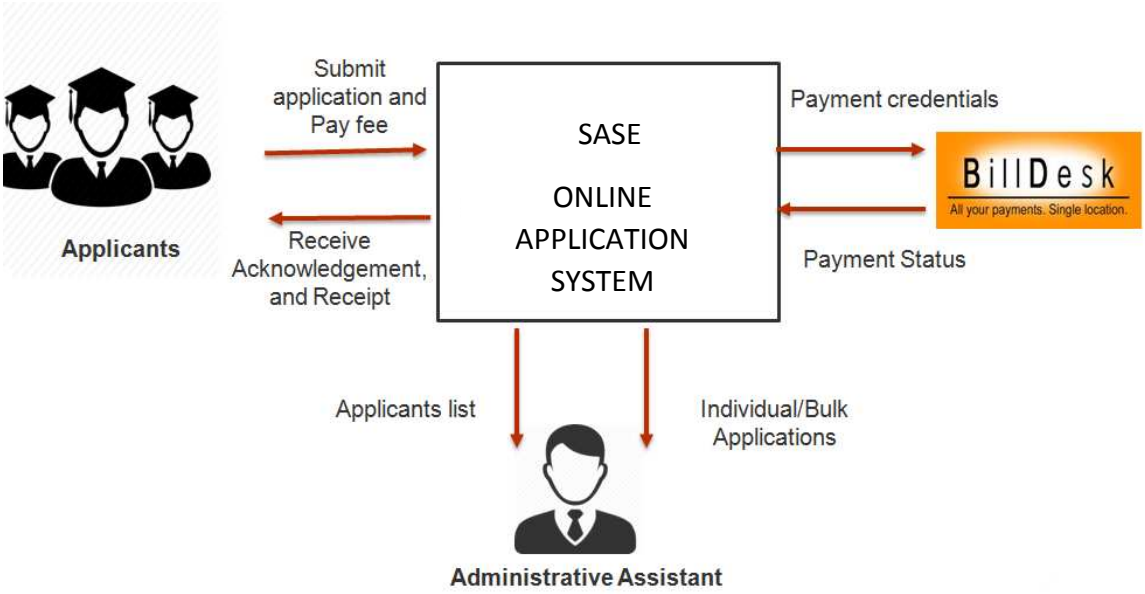
### Business Goals :

- To attract more applicants to apply to the institution by reducing their effort to apply.
- To reduce the effort of the administrative assistant in processing the application.

These goals can be realized by,

- Providing a secure way for the applicants to apply and pay for their application within minimum time.
- Providing a secured interface for the administrative assistant to manage and maintain the student records online.

### Context Diagram :




The main users of the system are,

The Applicant : The person who uses the system to submit his application to the institution and pay the application fee.

The Administrative Assistant : The person who uses the system to maintain and manage the student records for processing them.

## Part 2 - Architectural Drivers :

### High Level Functional Requirement :

The required functionality of the system include and is restricted to,

**For the applicant:**

- Register as user: The system shall provide an interface for new applicants to register in order to proceed to the application

- Login and logout: The system shall provide a secure authentication system which will permit only registered applicants to manage their applications
- Save and resume application: The system shall allow the applicant to save a partially filled application form and resume it later if the applicant wishes to logout and return later
- Upload Certificates: The system shall allow the applicant to upload his certificates in the desired format and constrained size.
- Preview and download: The system shall allow the applicant to preview completed application form and download the previewed application before proceeding to submit
- Submit and pay application fee: The system shall allow the applicant to submit the application and pay application fee in a secure manner. On successful payment, the system shall move the application for processing and shall provide him a reciept of payment.

**For SSN administrator:**

- Login and logout: The system shall provide a secure authentication system to the administrator to allow access to the administrator module of the application system
- Download application(s): The system shall allow the administrator to download applications***** individually or in bulk
- Sort / filter applicant list: The system shall provide sort and filter features that allows the administrator to generate the applicant list based on specific criteria
- Export reports: The system shall allow the administrator to export the generated applicant list into an excel report to facilitate manual processing, if required.
- Providing a separate interface for merging manually processed details of student applications with that of the students who applied through online application.

**Quality Attributes :**

**Usability : (P1)**

In order to enable more users to apply through the online application, the application should be user friendly and simple. The application should be designed in the way to satisfy the users mental model with proper affordances and signifiers like status and progress bars.

**Security: (P2)**

Confidential details like card numbers entered by the user are highly confidential and must be ensured that only the authorized payment gateway merchants would see it. The personal details and the scores of the applicant are also meant to be accessed only by the authorized personnel like an administrative assistant.

**Performance : (P3)**

The Applicant will access the application and will have various requests like submitting the form, saving the form and uploading the form. It is necessary that they are responded with a minimal latency time depending on the assigned priorities. These priorities are assigned based on the time the operation will take to complete.

**Availability: (P4)**

Any server component like a file system or a database may go down due to an overflow problem or an internal error. So it is necessary to detect this faults from the server end so that it can be reported to the user and also the recovery action like restart or switch to another drive in the file system takes place.

**Technical Constraints :**

- Programming Language – PHP
- Database – MySQL
- Apache HTTP Server
  - Intel Zeon E5405 Dual Processor 3.2 GHz
  - Server → Apache 2.2 (single server)

**Business Constraints :**

- Team members' time is limited to 12 hours per week.
- Project has to be completed before 15th April 2016.

**Quality Attribute Scenarios :**

| **Raw Quality Attribute** | Usability |
|---|---|
| **Stimulus** | Applicant opens the application and his previously entered data to be present in the application form. |
| **Source of the stimulus** | Applicant |
| **Environment** | Run Time |
| **Artifact** | System |
| **System Response** | The system automatically loads the data from the user's previously saved session |
| **Response Measure** | Time for the user to enter the same details every time he fills the form is saved. |

| **Raw Quality Attribute** | Usability |
|---|---|
| **Stimulus** | Applicant needs support in filling the forms and uploading the documents |
| **Source of the stimulus** | Applicant |
| **Environment** | Configure Time |
| **Artifact** | System |
| **System Response** | The System provides online help in the form of sequence of steps to be followed to complete the application filling process. |
| **Response Measure** | The user gains knowledge with respect to using the system. |

| **Raw Quality Attribute** | Usability |
|---|---|
| **Stimulus** | Applicant forgot his password and wants to reset it. |
| **Source of the stimulus** | Applicant |
| **Environment** | Run Time |
| **Artifact** | System |
| **System Response** | The system provides him a security question to be answered to reset his password without accessing his mail. |
| **Response Measure** | User is satisfied as his work with respect to access mails is saved.<br>Time for the user to enter the same details every time he fills the form is saved. |

| **Raw Quality Attribute** | Security |
|---|---|
| **Stimulus** | The administrative assistant wants to access the student's applications in a secured manner. |
| **Source of the stimulus** | Administrative assistant |
| **Environment** | Online |
| **Artifact** | Data within the system |
| **System Response** | System prompts for a login credentials to authorize the user. |
| **Response Measure** | On successful login, he will be redirected to applications window . |

| **Raw Quality Attribute** | Security |
|---|---|
| **Stimulus** | An illegitimate user trying to overload the server with his requests causing a "denial of service" for other legitimate users |
| **Source of the stimulus** | An attacker |
| **Environment** | Online |
| **Artifact** | Data within the system |
| **System Response** | System uses the pattern matcher to check for the burst requests of this type in the request queue |
| **Response Measure** | The system removes these requests and bans the IP from raising anymore requests for next 12 hours. |

| **Raw Quality Attribute** | Security |
|---|---|
| **Stimulus** | An attacker tries to perform a cross injection by inserting a javascript in the input fields. |
| **Source of the stimulus** | An attacker |
| **Environment** | Online |
| **Artifact** | Data within the system |
| **System Response** | The system uses the Input Validator and URL checker for any script embedded within it. |
| **Response Measure** | The system rejects those requests by not allowing the request to pass to the request queue. |

| **Raw Quality Attribute** | Security |
|---|---|
| **Stimulus** | An attacker has hacked his way into the credentials database and tries to access the login details stored in the database. |
| **Source of the stimulus** | An attacker |
| **Environment** | Online |
| **Artifact** | Data within the system |
| **System Response** | The system has encrypted the credential details before storing them in the database. |
| **Response Measure** | The system will give him the encrypted value which is of no use without the key. |

| **Raw Quality Attribute** | Security |
|---|---|
| **Stimulus** | An illegitimate user trying to gain access to the server through brute force combination of passwords. |

| | |
|---|---|
| **Source of the stimulus** | An attacker |
| **Environment** | Online |
| **Artifact** | Data within the system |
| **System Response** | The system detects a bruteforce attack and disables the userid for next 12 hours. |
| **Response Measure** | After 15 times of incorrect passwords, the user-id will be banned for 12 hours and both the applicant and administrator will be notified. |

| | |
|---|---|
| **Raw Quality Attribute** | Performance |
| **Stimulus** | The Applicant wants to save his current session and resume it in a future point of time. |
| **Source of the stimulus** | Applicant |
| **Environment** | Run Time |
| **Artifact** | System |
| **System Response** | All the details entered by the applicant are saved to the server and the applicant is notified regarding the status of the save operation performed. |
| **Response Measure** | The Applicant is notified of the status within 4 second of latency. |

| | |
|---|---|
| **Raw Quality Attribute** | Performance |
| **Stimulus** | The Applicant wants to submit the application and proceed to payment. |
| **Source of the stimulus** | Applicant |
| **Environment** | Run Time |
| **Artifact** | System |
| **System Response** | The Applicant details are saved to the server and the applicant is redirected to the payment window. |
| **Response Measure** | The Applicant is redirected to payment within 6 seconds of latency time. |

| | |
|---|---|
| **Raw Quality Attribute** | Performance |
| **Stimulus** | The Applicant wants to upload his marksheets and SOP |
| **Source of the stimulus** | Applicant |
| **Environment** | Run Time |
| **Artifact** | System |
| **System Response** | The applicant's documents are uploaded to the file system in the server after checking the file size |
| **Response Measure** | The Files will be uploaded within 10-15 seconds of latency depending upon its size. |

| **Raw Quality Attribute** | Availability |
|---|---|
| **Stimulus** | The user has uploaded some documents and no acknowledgement is received. |
| **Source of the stimulus** | Applicant |
| **Environment** | Run Time |
| **Artifact** | System |
| **System Response** | The Server Monitor has detected a fault with the file system and issues a restart command to it. |
| **Response Measure** | The user will be notified regarding the file system restart operation and time remaining to the file system to be back. |

| **Raw Quality Attribute** | Availability |
|---|---|
| **Stimulus** | The user wants to make payment and the redirection to the gateway is unsuccessful. |
| **Source of the stimulus** | Applicant |
| **Environment** | Run Time |
| **Artifact** | System |
| **System Response** | The Server Monitor has detected that the payment gateway is currently unavailable. |
| **Response Measure** | The user is notified regarding the breakdown within 3 seconds of his initiation of payment. |

**Part 3 - Architectural Description :**

**Mapping between Scenario-Concerns-Tactics and Elements :**

**Quality Attribute : Security**

| Concern | Concrete Scenario | Tactics | Elements |
|---|---|---|---|
| Availability | An illegitimate user trying to overload the server with his requests causing a "denial of service" for other legitimate users | Recovering from attack, identification and restoration | Pattern Matcher |
| Confidentiality | An attacker tries to perform a cross injection by inserting a javascript in the input fields. | Authorize users, Intrusion Detection | Input Validator, URL Checker. |
| Confidentiality | An attacker has hacked his way into the credentials database and tries to access the login details stored in it. | Resisting attack | SALT Encryptor, Decryptor |
| Confidentiality | An illegitimate user trying to gain access to the server through brute force combination of passwords. | Authorize users, Intrusion Detection | Input Validator, Security Log file |

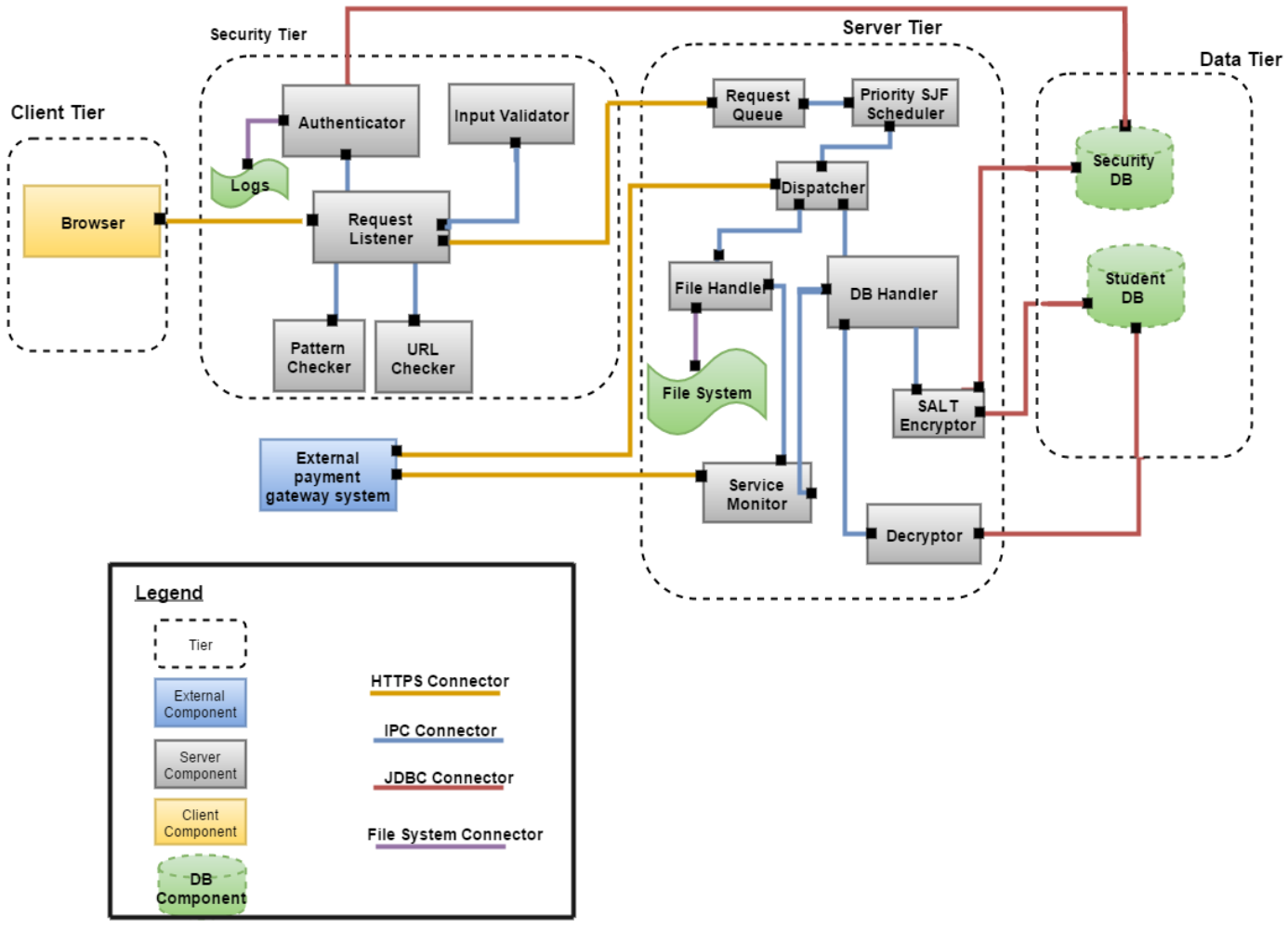
**Quality Attribute : Performance**

| Concern | Concrete Scenario | Tactics | Elements |
|---|---|---|---|
| Latency | An applicant wants to submit his application and proceed to payment. This redirection should be done within a latency of 6 seconds. | Resource arbitration | Priority SJF Scheduler, Cache |
| Latency | An applicant wants to save the data he has entered. This should be done within a latency of 4 seconds. | Resource arbitration | Priority SJF Scheduler, Cache |
| Latency | An applicant wants to upload his documents like mark sheets and Statement of Purpose etc. This should be done within 10-15 sec latency based on the size of the document. | Resource arbitration | Priority SJF Scheduler, Request Queue |

**Quality Attribute : Availability**

| Concern | Concrete Scenario | Tactics | Elements |
|---|---|---|---|
| Fault Detection | The user has uploaded some documents and yet there is no acknowledgement received. | Ping and Echo, Process Monitoring | Server monitor |
| Fault Detection | The user wants to save the data and is unable to do it due to a database crash . | Ping and Echo, Process Monitoring | Server Monitor |
| Fault Detection | The user wants to make payment and the redirection to the gateway is unsuccessful. | Raising Exceptions | Server Monitor |

**Dynamic View :**



**Element - Responsibility Catalogue :**

**Browser :** This is a component on the client side that requests applicant services and admin services from the server.

**Request Listener :** Server component that listens continuously for the service requests from the client through the browser.

**Pattern Checker :** Server component that checks for request burst patterns to detect Denial of Service attacks.

**Authenticator :** Authorizes the user to access the server for submitting the application or to access the students list by the administrative assistant.

**Input Validator :** Validates the form input for any executable scripts to prevent cross server injections from happening.

**URL Checker :** Validates theURL for any executable scripts to prevent cross server injections from happening.

**Priority SJF Scheduler :** A scheduler to schedule the requests that are in the request queue based on the time required to complete the task.

**Dispatcher :** Dispatches the request to the handlers based on the command from scheduler.

**File Handler :** Stores the documents provided by the user to the file system in the server.

**DB Handler :** Performs CURD (Create,Update,Read and Delete) operations on the database based on the user request.

**Server Monitor :** Monitors the key server components to detect and report faults.

**SALT Encryptor/Decryptor :** To decrypt the credentials and the card details before storing them to add an additional mechanism for securing the data from the Database.

**Security and Student DB :** Separate databases maintained for storing the data from the applicant and to store the login credentials of both the applicants and the administrative assistant.

**Design Rationale :**

The major design decisions , their effect on the quality attributes and their associated tradeoffs are documented as follows ,

**Decision :**

A separate Security tier was added to the basic three tier architecture pattern.

**Reason :**

The security checks against the various threats are performed here. If these requests threaten the security of our application then filtering them upfront in the separate security tier will be more secure than allowing them inside the server and then filtering. Allowing the requests inside the server tier will expose some of the server components to them. By adding a separate tier we are avoiding that exposure also.

**QAs and Trade-offs :**

Adding a separate tier will improve security by addressing the concerns of Confidentiality and Availability(Security Concern). This is done by scanning for various attacks as documented in the ER catalogue. This is a **sensitivity point** for security.

In one aspect, this will bring down performance in one aspect since a request has to pass through a separate tier before reaching the request queue in the server as it will increase the request propagation time and thereby increasing latency and decreasing performance of the system. This is a **trade off point** for performance.

But in another aspect it filters all unwanted requests upfront and hence only the proper requests will be in the queue. Since the number of requests are reduced, this might decrease the request service time in that sense and improve performance of the system.

Adding more components will raise the need to monitor them for faults by detecting and recovering from them. So this brings down availability. This is a **trade off point** for availability.

**Decision :**

Usage of HTTPS protocol over HTTP.

**Reason :**

Confidential data like card details are sent to the application server which has to be secured at any point of time in transmission and processing the request. So this gives the need to opt for HTTPS over the common HTTP protocol.

**QA's and Trade-Offs :**

HTTPS offers encryption of transmitted information using the SSL Certificates.

This additional encryption and Decryption will increase the request service time and thereby pulling down performance of the system. This is a **sensitivity point** for security.

But this performance is traded for addressing the security concern of credit/debit card data confidentiality as a slight delay in processing outweighs the devastations of exposing a credit card details to unauthorized person. This is a **trade off point** for performance.

**Decision :**

Introducing a Server Monitor components on the server side to detect and report faults.

**Reason :**

Designing for the feature to save and resume, uploading the documents and so on, we need to have a mechanism to detect faults if a file system component or a DB overflows or a payment gateway system is down and we should inform the user regarding that.

**QA's affected and Trade-Offs :**

This component will continuously monitor the key server components to detect and recover from fault. Hence it promotes availability. So it is a **sensitivity point** for availability.

Adding this component will introduce the overhead of maintaining the states of all these components and checking them continuously in periodic intervals will decrease performance. This is a **trade off point** for performance.

On a security perspective, this component is being shared by the key server components and hence if this component is being attacked by an outsider and in case if he gains control over the component, then he will have access to all the server components that share this monitor. This is a **trade off point** for security

But Security is traded off for availability here is because odds of that kind of an attack happening overcoming all our security barriers is remote when compared to a fault occurring in a server component.

**Decision :**

Adding a priority Shortest Job First priority scheduler instead of a default scheduler available in PHP

**Reason :**

Given that the server has to address several type of requests like saving data into the database, uploading files to the file system, redirection to the payment gateway and so on, these tasks has to be scheduled to follow an order and optimize the servicing of requests by the server.

**QA's affected and Trade-Offs :**

Using a basic FCFS scheduler might introduce situations like a person may be uploading 10 documents but another person who wants to just save his details to the database will have to wait for all the time.

So, to avoid this kind of situations we chose to go with a preemptive SJF scheduler to assign priorities based on the time required to complete the task.

This will improve the performance by scheduling the arriving requests based on assigning priorities and also the users wait time is proportional to their the time required to complete the task . So this will satisfy their mental models promoting usability. This is a **sensitivity point** for performance.

As earlier, adding this component raises the need to monitor this component for detecting and recovering from faults. So this decreases availability. This is a **trade off point** for availability.

**Decision**

Having two separate DB's for storing credentials and student data.

**Reason :**

Login functionality and Student details entry and retrieval are two separate tasks. Introducing them in a separate DB will promote concurrency in processing the DB requests by the database handler.

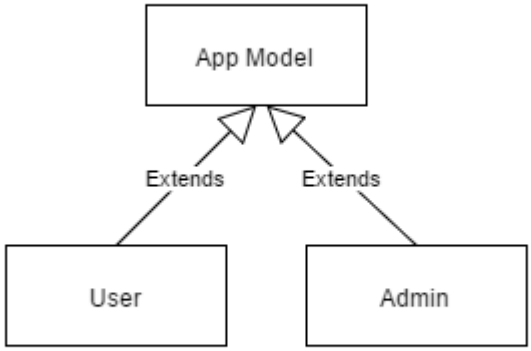
**QA's affected and Trade-Offs :**

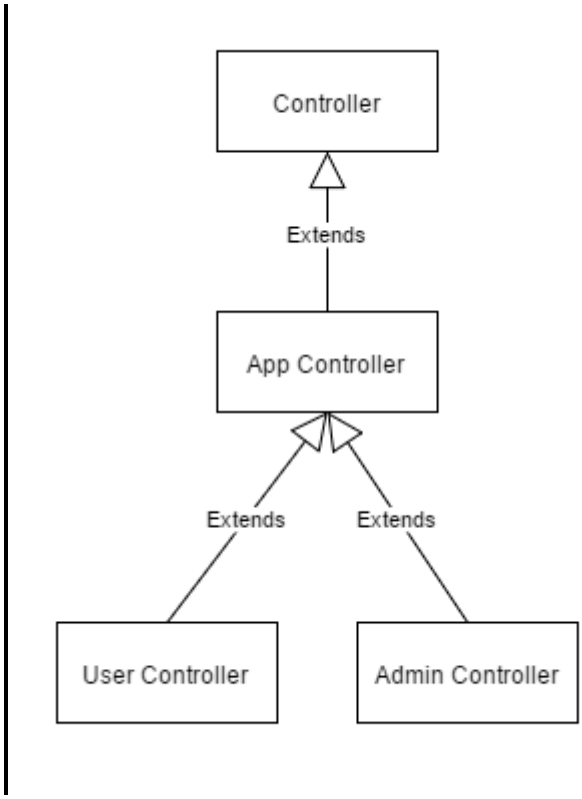
Since adding this component to the architecture promotes concurrency, this will improve the request service rates and thereby performance of the system. This is a **sensitivity point** for performance.

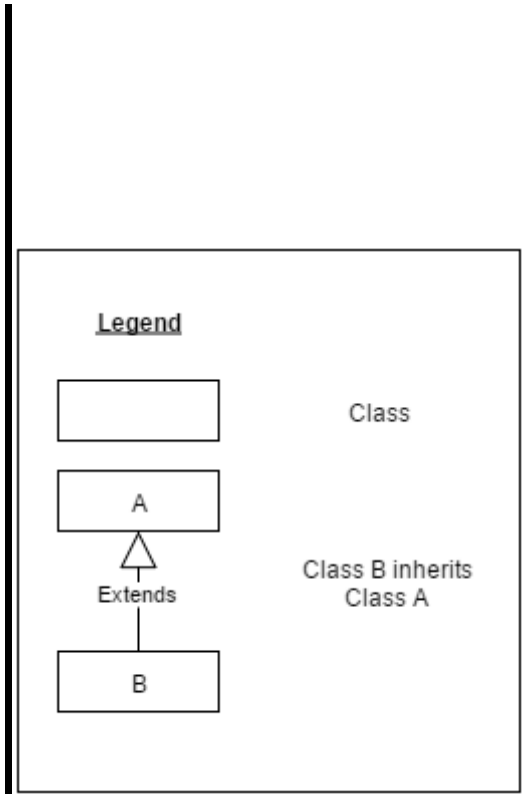
Having the login credentials encrypted and stored in a separate database also promotes security, because even if an attacker gains access into one database, the other one is safe from SQL injection. This is a **sensitivity point** for security.

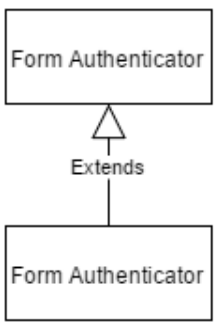
This will also improve maintainability , but that is not considered here as it is not our quality attribute requirement.

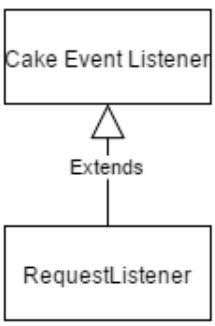
**Module View :**

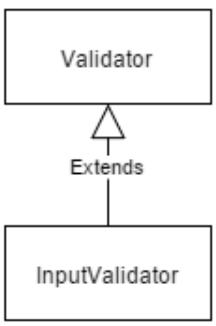

App Model
Extends
Extends
User
Admin

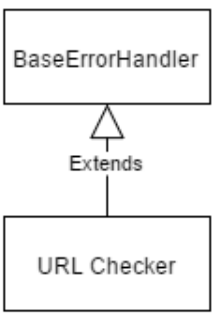

Controller
Extends
App Controller
Extends
Extends
User Controller
Admin Controller


Legend
Class
A
Extends
B
Class B inherits
Class A


Form Authenticator
Extends
Form Authenticator


Cake Event Listener
Extends
RequestListener


Validator
Extends
InputValidator


BaseErrorHandler
Extends
URL Checker

**Element- Responsibility Catalogue :**

**class AppModel**

AppModel class is the default application model class in CakePHP framework. It provides methods for accessing, adding, modifying data from the database as well as for interaction with other models. This class is the parent class for several inheriting models throughout the application such as class User and class Admin.

**class User extends AppModel**
**class Admin extends AppModel**

These classes will contain the custom validation rules for user functions such as unique email IDs during user registration and other registration rules before storing the successfully registered user credentials onto the database..

**class AppController extends Controller**

The AppController class is the parent class to all of the application's controllers. Controller actions in this class are responsible for converting the request parameters into a response for the browser/user making the request.

**class UsersController extends AppController**

UsersController class provides methods to redirects the user to the right page after login or to the page the user was previously in, before the session expired. In addition, this class also contains functions for login, logout and all functional aspects of filling, editing and saving the application.

**class AdminsController extends AppController**

This class contains methods for redirecting the administrator to the right page after logging in, as well as the methods for the functional aspects of administrator functions such as viewing and downloading the list of applicants.

**class Authenticator extends FormAuthenticate**

This class extends and customizes the functionality of salt-encryption method to authenticate the user by matching his credentials and allows access to the user.

**class RequestListener extends CakeEventListener**

This class provides methods to register the callbacks for an event. This is done by implementing the CakeEventListener interface and extending it to register and customize the parameters of the events for which the callbacks are to be registered.

**class InputpValidator extends Validator**

The Validator class in CakePHP provides features to build validators that can validate arbitrary arrays of data with ease. The InputValidator class defines the methods to validate the input data for any scripts available that could lead to an SQL injection.

**class Dispatcher extends DispatcherFilter**

This class converts requests into controller actions. It uses the the dispatched request to locate and load the correct controller. The methods in DispatcherFilter class have the ability to alter the request or response as needed before it is handled by a controller, in order to ensure that invalid or incorrect requests do not get a response.

**class Scheduler**

This class provides methods for prioritizing shortest jobs first from the request queue and sending them to the dispatcher so that

**class FileHandler extends File**

This class contains methods for various file operations such as writing data to files (PDF for preview by the student user) as well as reading the data from the files and displaying them to the user (administrator).

**class PatternChecker**

This class is used to detect specific patterns a hacker may use in order to gain access to the application. From the apache web server access log, this class would identify when someone

> made multiple requests in less than second or accepted time frame

> accessed secure or login page multiple times in a one-minute window

> accessed nonexistent pages using different query parameters or path

**class UrlChecker extends BaseErrorHandler**

This class is used to prevent direct execution of webpages when the user attempts to use direct URLs to go to a webpage without being authorized to do so. Cross site request forgery (CSRF) is a technique commonly used by hackers to capture and replay a previous request without being authorized. This class extends and adds to the functionality of BaseErrorHandler, which returns an error when a CSRF exception occurs.

**class ServiceMonitor**

This class is used to redirect the user to the payment gateway, which returns the request to the dispatcher once the payment transaction has been completed.

**Part 4 - Architectural Analysis :**

As suggested by the companion guide, due to time constraints, only some of the steps in Architectural Trade off Analysis Method (ATAM) is taken into consideration and followed for analyzing the above architecture.

Steps 1-3 :

Presenting the business goal, quality attributes, their mapping and architecture are presented earlier in the same document. So they are not repeated here.

**Construction of Utility Tree :**

| Quality Attribute | Refinement | Analysis Result scenarios |
|---|---|---|
| Security | Availability | An illegitimate user trying to overload the server with his requests causing a "denial of service" for other legitimate users and the request from that IP is banned for the next 12 hours. [H,M] |
| | Confidentiality | An attacker tries to perform a cross injection by inserting a javascript in the input fields and the input request is rejected after the script is detected by input checker. [H,M] |
| | Confidentiality | An attacker has hacked his way into the credentials database and tries to access the login details stored in it. The data is SALT encrypted and is of no use to the hacker. |
| | Confidentiality | An illegitimate user trying to gain access to the server through brute force combination of passwords. After 15 wrong passwords to the same username, the user-id gets blocked. [H,M] |
| Performance | Latency | An applicant wants to submit his application and proceed to payment. This redirection should be done within a latency of 6 seconds.[M,M] |
| | Latency | An applicant wants to save the data he has entered. This should |

| | | |
|---|---|---|
| | | be done within a latency of 4 seconds. [M,M] |
| | Latency | An applicant wants to upload his documents like mark sheets and Statement of Purpose etc. This should be done within 10-15 sec latency based on the size of the document. [M,M] |
| Availability | Fault Detection | The user has uploaded some documents and yet there is no acknowledgement received. This fault is detected and reported to the applicant by the service monitor. [M,M] |
| | Fault Detection | The user wants to save the data and is unable to do it due to a database crash. This fault is detected and reported to the applicant by the service monitor. [M,M] |
| | Fault Detection | The user wants to make payment and the redirection to the gateway is unsuccessful. This fault is detected and reported to the applicant by the service monitor. [M,M] |

For each scenario in the above table, priority rankings are annotated based on the decision makers ranking, in our context as a team. The first of the ordered pair indicates the importance of capability and indicates the architect's estimation of difficulty in achieving it.

**Identification of Sensitivity Points and Trade-Off points :**

These points are already identified in Part 3 - pg no 11 to 14 while articulating the design decisions, tradeoffs and impact on quality attributes. So they are not repeated again.

**Check list of ATAM Outputs :**

- ✓ Presentation of Business Goals
- ✓ Presentation Of Architecture
- ✓ Quality requirements in terms of Scenario
- ✓ Mapping of Architectural Decisions to Quality scenarios.
- ✓ Create a Utility Tree
- ✓ Identified Sensitivity Points
- ✓ Identified Trade-off Points

**References :**

All the approaches followed in this document are used from,

Bass, Len; Clements, Paul; & Kazman, Rick. ***Software Architecture in Practice***, 2nd ed. Reading, MA: Addison-Wesley, 2003.

Clements, Paul; Bachmann, Felix; Bass, Len; Garlan, David; Ivers, James; Little, Reed; Nord, Robert; & Stafford, Judith A. ***Documenting Software Architectures: Views and Beyond*.** Reading, MA: Addison-Wesley, 2003.