AVIATION INDEX PROJECT DESIGN

Jesse Dalton

CST-451 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Professor Michael Landreth

May 4, 2025

THIS IS MY OWN WORK

TABLE OF CONTENTS

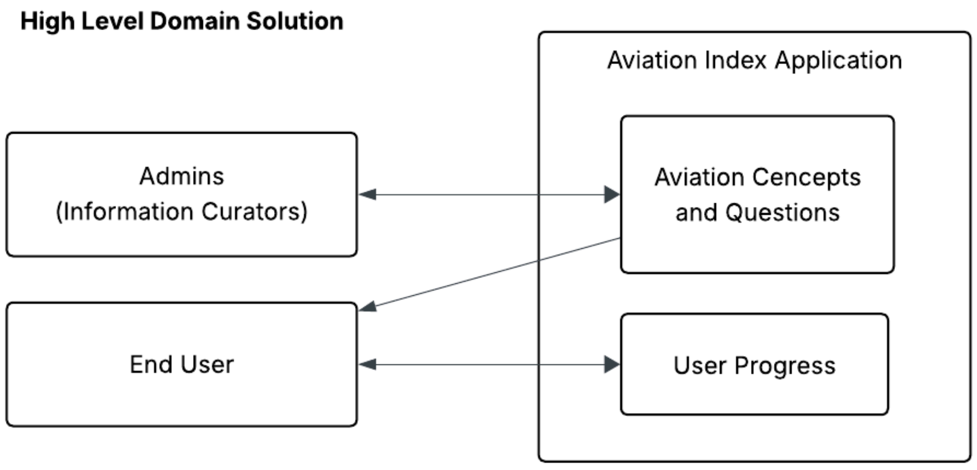
DELIVERABLE ACCEPTANCE LOG

ID	Deliverable Description	Comments	Evaluator	Status	Date of Decision
1	Project Proposal	Previously submitted via Halo			
2	Project Requirements	Previously submitted via Halo			
3	Project Design	This document			
4	Data Dictionary	Submitted via Halo			

DESIGN OVERVIEW

Aviation Index is a comprehensive web-based study tool designed to support pilots throughout their initial training and their professional careers. The goal of the application is to provide a comprehensive, well-organized knowledge base of aviation concepts, allowing users to interact with small information points through an intuitive interface.

The application will allow admins to organize and curate the aviation concepts and questions. Regular users can then view this information under different filters and cycle through flash-card-style questions to test their understanding and commit the concepts to memory. The application will offer ways to track and report user progress as they study.


High Level Domain Solution
Aviation Index Application
Admins (Information Curators)
Aviation Cencepts and Questions
End User
User Progress

DETAILED HIGH-LEVEL SOLUTION DESIGN

Aviation Index has 4 major Project Objectives (see Project Proposal):

1. Persist a Centralized Knowledge Base of Aviation Information
2. Provide an Efficient Study Tool Interface to interact with this information
3. Allow Users to customize their learning experience
4. Maintain Accurate and Reliable information over time

Aviation Index is an N-tier architecture, web-based software application that meets the previously mentioned Project Objectives.

The application is written in the Java Programming Language, using the Spring Framework and associated idiomatic technology stack, which includes the Thymeleaf Templating Engine and Spring Boot Web, Spring Boot Security, and Spring Boot JPA libraries. Data will be persisted in a MySQL database, which is accessible via the JPA library with some basic configuration.

ARCHITECTURE LAYERS

Layer	Description	Technologies
Models	Simple objects that encapsulate the application domain	Spring Boot JPA
User Interface	Public and authenticated web pages to interface with the server	HTML, CSS, JavaScript, Thymeleaf
Network	Allow communication between the client and the server across the web	Spring Boot Web Spring Boot Security
Service	Application business rules, logic, functionality, and processes	Spring
Persistence	Persist domain data in a relational database	Spring Boot JPA MySQL

DOMAIN MODELS

*** Technical Documents provided in Detailed Technical Design:**

- Model UML Class Diagrams
- ER Diagram
- DDL Script
- Data Dictionary

TOPICS

Aviation Index organizes all aviation knowledge content into categories called Topics. Every Topic can contain a list of other Topics, called Subtopics. This recursive relationship creates a hierarchical system that facilitates organization and navigation of the the system's content.

The hierarchical relationship is transitive. A Subtopic of a Subtopic is also considered a Subtopic of the original Topic. For example, if Topic A contains Subtopic B, and Topic B contains Subtopic C, then Topic C is also a Subtopic of Topic A.

QUESTIONS

The most atomic unit of knowledge content in the application is a Question. Each Question encapsulates a question-answer pair. It invites the user to identify or describe a bite-sized aviation concept, then provides the corresponding answer.

Every Question must belong to a Topic. Because Topics are hierarchical and transitive, each Question is also implicitly associated with all parent Topics of the Topic it directly belongs to. For example, if Topic A contains Subtopic B, and Topic B contains Questions X, Y, and Z, then Questions X, Y, and Z are also considered part of Topic A.

USERS

Users can create an account to use the application. A User represents the real-world individual and contains both identifying and authentication information, as well as references to other entities that describe their interaction with the system.

Users are identified by a username and authenticated by a password. Spring Security uses these credentials to manage secure user and admin sessions.

NOTE: User Passwords are encrypted using BCrypt before being stored in the database.

A major domain concept of Aviation Index is the Study Session, although it is not modeled by its own entity. Instead, details that describe the current User’s Study Session are stored directly in the User model. These references act as filters that determine which content the application presents to the User. These parameters span all contexts of the application. For example, if a User has a reference to Topic A, they will only see Questions associated with Topic A.

USER QUESTIONS

UserQuestions represent a User’s individual progress with respect to each Question. They do not contain information about the User or Question. They are simply a record of the relationship between the two entities.

Each UserQuestion contains a status, which can be set to “new”, “learning”, “mastered”, or “focus”, allowing the User to track their personal progress.

UserQuestions can be flagged as “active”, which allows the application to internally determine which Questions currently belong to the User’s Study Session. When an update is made to a User’s Study Session, the application silently adjusts the active flags based on the new study filters.

USER INTERFACE LAYER

Aviation Index provides two clients for interacting with the domain content from different perspectives. The User Client is a set of web pages that allow potential users to learn about the application and allow current users to use the application. Although authenticated users have read access to most of the aviation content, they are unable to make changes. Write access is only granted to users for making changes to their progress in relation to studying the content.

The Admin Client is a set of web pages that allow admins to curate the aviation information provided by the application. They have full read/write access to all general aviation content provided by the application, however, they are unable to make changes to personal user information, such as profiles and progress.

NOTE: Admins are considered a subclass of users and can access all content made available to regular users.

All client pages are generated using the Thymeleaf Templating Engine.

*** Technical Documents provided in Detailed Technical Design:**

- Sitemap
- All Wireframes

PUBLIC PAGES

Pages accessible to any public user include Home, Login, and Register. The Home page, or landing page, provides context for potential users of the application's scope, purpose, intended audience, benefits, and relevance. Login and Register are typical user management interfaces.

Page	**URI**	**Description**	**Wireframe ***	**Models ***
Home	/	Landing page that describes Aviation Index and markets to potential users	Home	none
Login	/login	Login form for users to authenticate	Login	User
Register	/register	Register form to create new user	Register	User

AUTHENTICATED USER PAGES

Once authenticated, a user will access the ‘bread and butter’ of the application, a set of pages designed to organize, filter, and present the aviation information. This includes the Topics, Questions, Filters, and Study pages.

The Study page serves a single question, dynamically selected according to the user’s parameters, and allows the user to study the question. Designed like a flashcard, the Study page initially hides the answer and allows the user to display the answer when ready. The user can flag the question as ‘New’, ‘Focus’, ‘Learning’, or ‘Mastered’. This parameter is one of the filters that can be adjusted on the Filters page and applied to the question pool from which the Study page pulls each question.

A final page, Reports, can be used to view the user’s progress as they view and flag questions.

Page	**URI**	**Description**	**Wireframe ***	**Models ***
Topics	/topics/{id}	View a single topic and associated subtopics and questions	Topics	Topic Question
Questions	/questions	View all questions returned by a search	Questions	Question
Filters	/filters	Session settings page for users to adjust study parameters	Filters	User
Study	/study	Study a single question and update personal progress	Study	Question UserQuestion
Reports	/reports	View various reports of personal progress across all domain content	Reports	User UserQuestion
Profile	/profile	Update user profile	Profile	User

AUTHENTICATED ADMIN PAGES

Once authenticated, an admin will access a main dashboard page and a set of utility pages that allow them to curate the knowledge content of the application. Admins will create, edit, and delete topics and questions. They will also organize topics in a hierarchy that facilitates searching for specific content. They will assign questions to topics. These groupings will allow users to adjust the scope of their study sessions.

Page	**URI**	**Description**	**Wireframe ***	**Models ***
Admins	/admins	Dashboard page that allows admins to curate aviation knowledge content	Admins	Topic Question
Add Topic	/topics/add	Form to add a new topic	Topic-Add	Topic
Edit Topic	/topics/edit	Form to edit an existing topic	Topic-Edit	Topic
Add Question	/questions/add	Form to add a new question	Question-Add	Question
Edit Question	/questions/edit	Form to edit an existing question	Question-Edit	Question

NETWORK LAYER

Aviation Index is ready to be deployed to a public hosting platform. Using Spring Boot Web, HTTP requests are received and processed. Corresponding HTTP responses are also generated and served.

At this point, all responses are web pages generated using Thymeleaf. Future changes to the application might include an API that allows general access to the application services. This would allow support for multiple clients in different environments. For now, Aviation Index is designed to only function as a simple web app.

This means that every response served by Aviation Index will be a page of HTML with corresponding CSS and JavaScript if necessary.

*** Technical Documents provided in Detailed Technical Design:**

- Network Layer UML Class Diagram

SECURITY

Aviation Index uses Spring Security to protect sensitive data and manage authenticated user sessions. User credentials are secured using BCrypt hashing, and only non-sensitive fields are ever exposed to the client. The application restricts access to protected endpoints based on user roles, ensuring that only authorized users can perform administrative actions or access personalized data. Spring Security also manages session state, enabling users to log in and interact securely with the application without repeatedly re-authenticating. This setup provides strong confidentiality and integrity while keeping the user experience seamless.

SERVICE LAYER

Aviation Index implements the application business rules and functionality in the service layer. This is where the relationships between entities are defined and also where the logic is developed that allows users to leverage the service in a meaningful way.

*** Technical Documents provided in Detailed Technical Design:**

- Admin Processes Flowchart
- User Processes Flowchart

PERSISTENCE LAYER

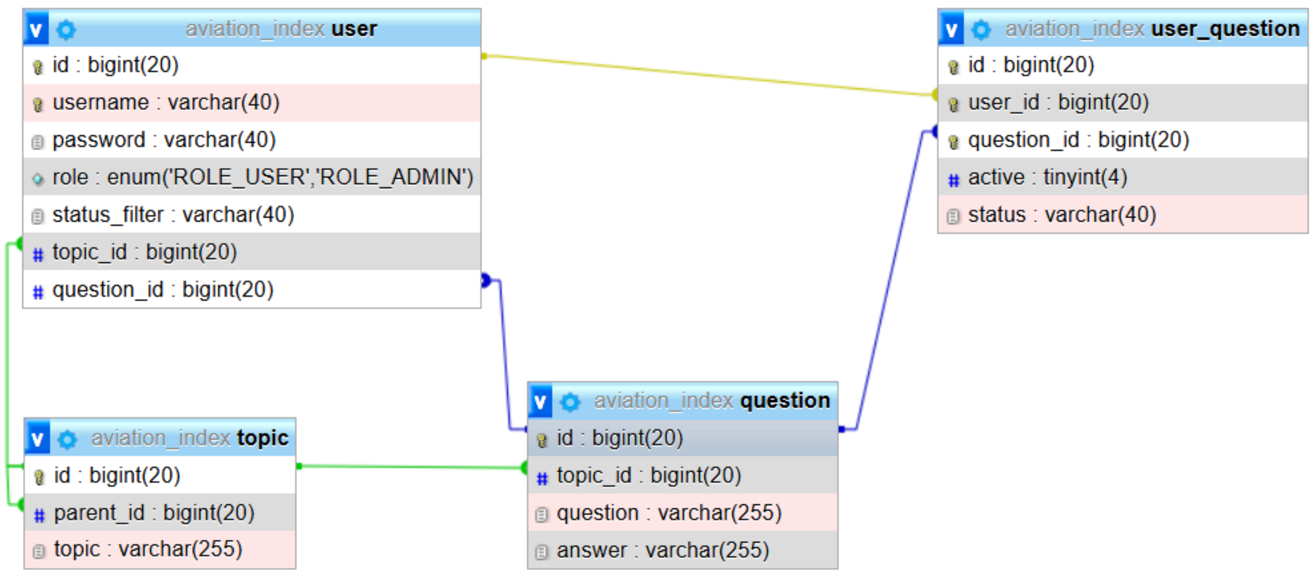
Aviation Index is designed using JPA and attaching to a MySQL database. This database must be intialized using the provided DDL script. JPA is a flexible library that can be configured to work with many common database, so the application is not limited to MySQL.

DETAILED TECHNICAL DESIGN

GENERAL TECHNICAL APPROACH

The application is written in the Java Programming Language, using the Spring Framework and associated idiomatic technology stack, which includes the Thymeleaf Templating Engine and Spring Boot Web, Spring Boot Security, and Spring Boot JPA libraries. Data will be persisted in a MySQL database, which is accessible via the JPA library with some basic configuration.

DATABASE ER DIAGRAM

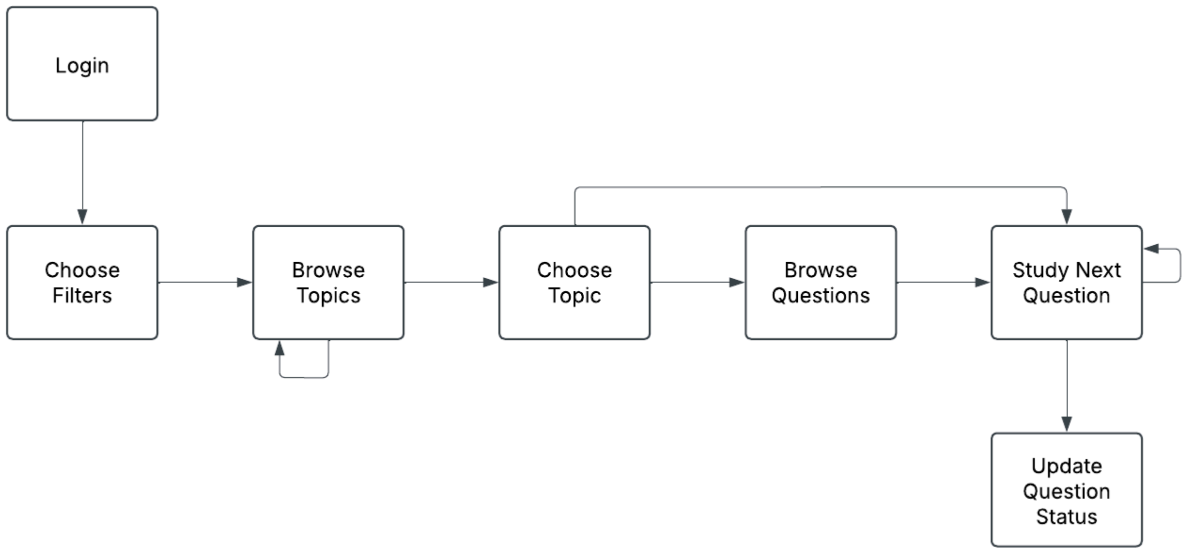


DATABASE DDL SCRIPT

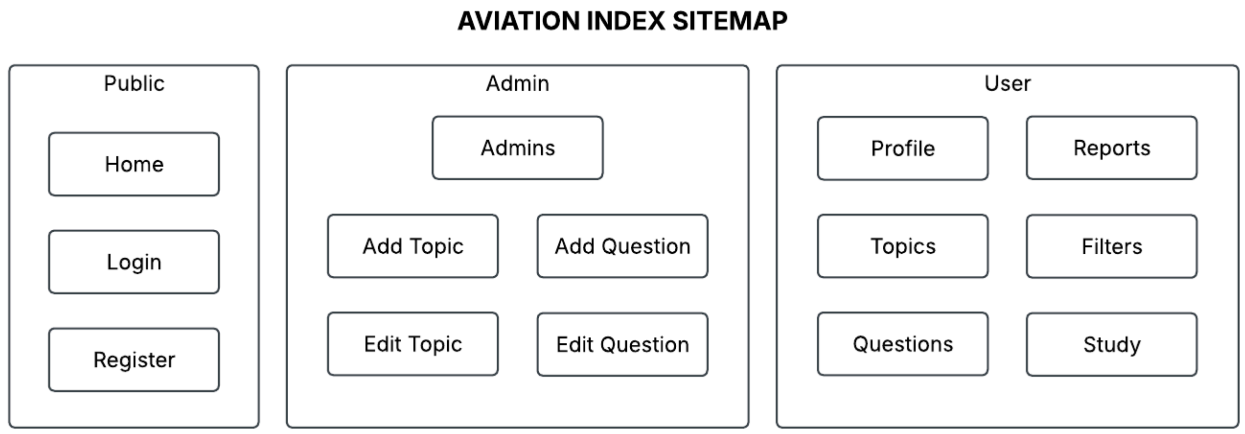
```
DROP DATABASE IF EXISTS aviation_index;
CREATE DATABASE aviation_index;
USE aviation_index;
CREATE TABLE user (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(40) NOT NULL UNIQUE,
    password VARCHAR(40) NOT NULL,
    role ENUM("ROLE_USER", "ROLE_ADMIN") NOT NULL,
    status_filter VARCHAR(40),
    topic_id BIGINT,
    question_id BIGINT,
    FOREIGN KEY (topic_id) REFERENCES topic(id),
    FOREIGN KEY (question_id) REFERENCES question(id)
);
CREATE TABLE topic (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    parent_id BIGINT,
    topic VARCHAR(255) NOT NULL,
    FOREIGN KEY (parent_id) REFERENCES topic(id)
);
CREATE TABLE question (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    topic_id BIGINT,
    question VARCHAR(255) NOT NULL,
    answer VARCHAR(255) NOT NULL,
    FOREIGN KEY (topic_id) REFERENCES topic(id)
);
CREATE TABLE user_question (
    id BIGINT PRIMARY KEY AUTO_INCREMENT,
    user_id BIGINT NOT NULL,
    question_id BIGINT NOT NULL,
    active TINYINT NOT NULL,
    status VARCHAR(40),
    FOREIGN KEY (user_id) REFERENCES user(id) ON DELETE CASCADE,
    FOREIGN KEY (question_id) REFERENCES question(id) ON DELETE CASCADE,
    UNIQUE (user_id, question_id)
);
```

FLOWCHARTS AND PROCESSES

User Process Flow

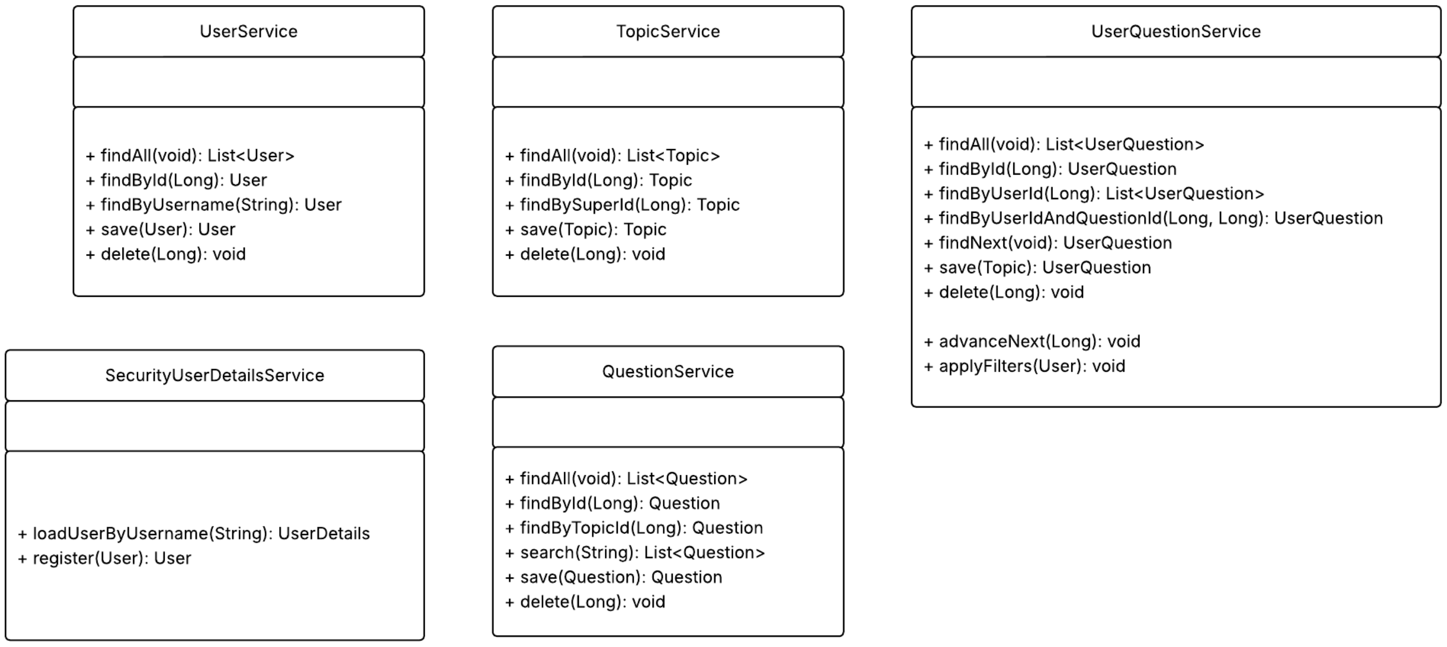


SITEMAP


AVIATION INDEX SITEMAP
Public
Home
Login
Register
Admin
Admins
Add Topic
Add Question
Edit Topic
Edit Question
User
Profile
Reports
Topics
Filters
Questions
Study

UML DIAGRAMS

Service Layer Classes


UserService
+ findAll(void): List<User>
+ findById(Long): User
+ findByUsername(String): User
+ save(User): User
+ delete(Long): void
TopicService
+ findAll(void): List<Topic>
+ findById(Long): Topic
+ findBySuperId(Long): Topic
+ save(Topic): Topic
+ delete(Long): void
UserQuestionService
+ findAll(void): List<UserQuestion>
+ findById(Long): UserQuestion
+ findByUserId(Long): List<UserQuestion>
+ findByUserIdAndQuestionId(Long, Long): UserQuestion
+ findNext(void): UserQuestion
+ save(Topic): UserQuestion
+ delete(Long): void
+ advanceNext(Long): void
+ applyFilters(User): void
SecurityUserDetailsService
+ loadUserByUsername(String): UserDetails
+ register(User): User
QuestionService
+ findAll(void): List<Question>
+ findById(Long): Question
+ findByTopicId(Long): Question
+ search(String): List<Question>
+ save(Question): Question
+ delete(Long): void

APPENDIX A - TECHNICAL ISSUE AND RISK LOG

APPENDIX B - REFERENCES

APPENDIX C - EXTERNAL RESOURCES

GitHub URL

https://github.com/jmdalton0/aviation-index