1 Module Guide

1.1 Hardware Hiding Modules

Secrets: The data structure and algorithm used to implement the virtual hardware.

Services: Serves as a virtual hardware used by the rest of the system. This module provides the interface between the hardware and the software. So, the system can use it to display outputs or to accept inputs.

Implemented by: OS

1.2 Behaviour Hiding Modules

1.2.1 User Interface Module (M1)

Secrets: How to allow users to go through the process of buying and selling textbooks

Services: Provides the user interface of the application

Implemented by: Book Bazar

1.2.2 Post HTTP Handler Module (M2)

Secrets: How to respond to REST API requests for posts

Services: Provides the ability to create, get, update, and delete posts

Implemented by: Book Bazar

1.2.3 Book HTTP Handler Module (M3)

Secrets: How to respond to REST API requests for books

Services: Provides the ability to get information for a book, including posts for it

Implemented by: Book Bazar

1.2.4 Course HTTP Handler Module (M4)

Secrets: How to respond to REST API requests for courses

Services: Provides the ability to get information for a course, including books and posts for it

Implemented by: Book Bazar

1.2.5 Auth HTTP Handler Module (M5)

Secrets: How to respond to REST API requests for authentication

Services: Provides the ability to create users, authenticate them, and logout

Implemented by: Book Bazar

1.2.6 User HTTP Handler Module (M6)

Secrets: How to respond to REST API requests for users

Services: Provides the ability to get, update, and delete users

Implemented by: Book Bazar

1.2.7 User Service Module (M7)

Secrets: How to get and modify user information

Services: Provides the ability to retrieve and modify stored user information

Implemented by: Book Bazar

1.2.8 Course Service Module (M8)

Secrets: How to get course information

Services: Provides the ability to retrieve courses and their related information

Implemented by: Book Bazar

1.2.9 Post Service Module (M9)

Secrets: How to get and modify post information

Services: Provides the ability to retrieve and modify stored post information

Implemented by: Book Bazar

1.2.10 Book Service Module (M10)

Secrets: How to get book information

Services: Provides the ability to retrieve stored book information

Implemented by: Book Bazar

1.2.11 Session Service Module (M11)

Secrets: How to get and modify session information

Services: Provides the ability to retrieve and modify stored session information

Implemented by: Book Bazar

1.2.12 Magic Link Service Module (M12)

Secrets: How to send and consume magic links

Services: Provides the ability to send magic link emails and consume them to create a new session

Implemented by: Book Bazar

1.2.13 Image Service Module (M13)

Secrets: How to store an image

Services: Provides method to store an image in cloud storage for retrieval with a URL

Implemented by: Book Bazar

1.2.14 Campus Store Scrapper Module (M14)

Secrets: How to index the campus store textbook library

Services: Provides the ability to retrieve and store textbook and course information from the campus store

Implemented by: Book Bazar

1.2.15 Prisma Client Module (M15)

Secrets: How to read from and write to the database

Services: Provides the ability to retrieve and modify information stored in a database

Implemented by: @prisma/client

1.2.16 AWS SDK Module (M16)

Secrets: How to interact with Amazon Web Services (AWS)

Services: Provides the ability to use and manage AWS services. This project is mainly interested in the S3 API

Implemented by: aws-sdk

1.2.17 Sendgrid Mail Module (M17)

Secrets: How to interact with the Sendgrid service

Services: Provides the send emails with the Sendgrid email API

Implemented by: @sendgrid/mail

1.2.18 Algolia Search Module (M18)

Secrets: How to interact with the Algolia search service

Services: Provides method to index and search for courses and books in Algolia

Implemented by: algoliasearch

1.2.19 Google Books Search Module (M19)

Secrets: How to interact with the Google Books search service

Services: Provides method to search for and retrieve books from Google Books

Implemented by: googleapis

1.3 Software Decision Modules

1.3.1 Tokens Module (M20)

Secrets: How to tokens are created and hashed

Services: Provides methods to create and hash cryptographically secure and random tokens

Implemented by: Book Bazar

1.3.2 Environment Module (M21)

Secrets: Environment variable constants

Services: Provides various environment variable constants

Implemented by: Book Bazar

1.3.3 Session Cookie Module (M22)

Secrets: How to create and delete session cookies

Services: Provides methods to create and delete HTTP session cookies

Implemented by: Book Bazar

1.3.4 User Helpers Module (M23)

Secrets: How to interact with user data

Services: Provides methods to perform common tasks with user objects

Implemented by: Book Bazar

2 Traceability Matrix

Requirement	Modules
FR1	M1, M2, M9
FR2	M1, M2, M9
FR3	M1, M2, M9
FR4	M1, M10, M14
FR5	M1, M4, M8
FR6	M1, M3, M10
FR7	M1
FR8	M1, M6
FR9	M1, M2, M3, M4, M5, M6, M7
FR10	M1, M3, M10, M14
FR11	M1, M5, M7, M12
FR12	M1, M6, M7
FR13	M1
FR14	M1, M6

Table 1: Traceability Matrix

3 Uses Hierarchy Between Modules

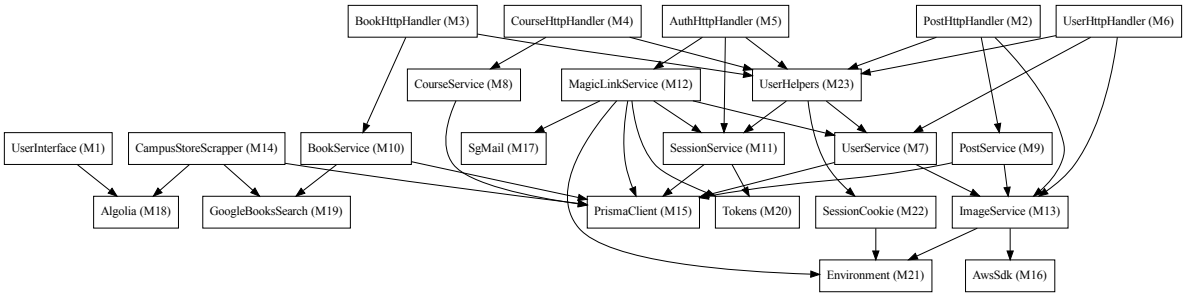

Figure 1: Uses Hierarchy Between Modules

4 Anticipated Changes

4.1 Likely Changes

AC1: The frontend design for the application will change as more feedback about it's usability is received

AC2: The authentication system will be updated to make magic links sent to user emails more simple and secure to handle

AC3: The campus book store scraper will be updated if the campus store website source code change

AC4: Calls to the Google Books API will be updated according to API or data changes

AC5: Calls to the Algolia API will be updated according to API or data changes

AC6: Calls to the Sendgrid API will be updated according to API or data changes

AC7: The image cloud storage provider will be changed based on system needs

4.1.1 Traceability matrix

Requirement	Modules
AC1	M1
AC2	M5, M11, M12
AC3	M14
AC4	M19
AC5	M18
AC6	M17
AC7	M13

4.2 Unlikely Changes

UC1: The frontend design for the application will change as more feedback about it's usability is received

UC2: The authentication system will be updated to make magic links sent to user emails more simple and secure to handle

UC3: The campus book store scraper will be updated if the campus store website source code change

UC4: Calls to the Google Books API will be updated according to API or data changes

UC5: Calls to the Algolia API will be updated according to API or data changes

UC6: Calls to the Sendgrid API will be updated according to API or data changes

UC7: The image cloud storage provider will be changed based on system needs

User Interface Module (M1)

Module

UserInterface (/pages)

Uses

Algolia

Syntax & Semantics

Implements user interface defined at `https://www.figma.com/file/FS5U5ssCSPctWH41VhGIsn/Book-Bazar` and by UML state diagram below:

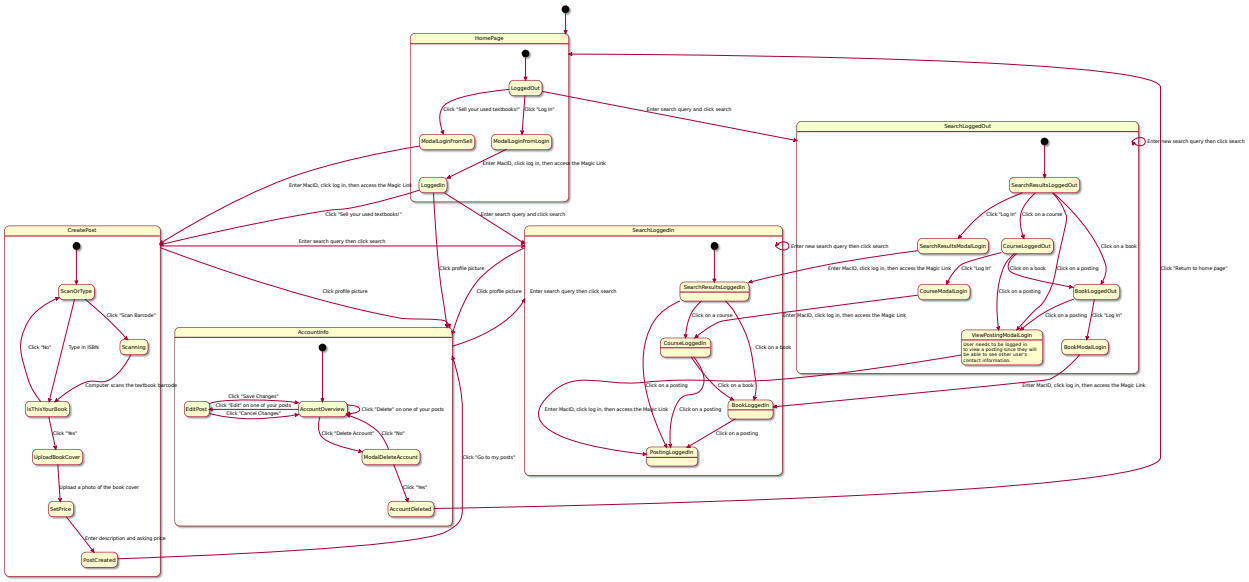

Figure 2: User Interface UML state diagram

Post HTTP Handler Module (M2)

Module

PostHttpHandler (/pages/api/post)

Uses

PostService
ImageService
UserHelpers

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
handler	NextApiRequest, NextApiResponse	N/A	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

handler(req, res):

- Implements OpenAPI spec defined at `https://bookbazar.me/api/swagger#/Post`

Book HTTP Handler Module (M3)

Module

BookHttpHandler (/pages/api/book)

Uses

BookService
UserHelpers

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
handler	NextApiRequest, NextApiResponse	N/A	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

handler(req, res):

- Implements OpenAPI spec defined at `https://bookbazar.me/api/swagger#/Book`

Course HTTP Handler Module (M4)

Module

CourseHttpHandler (/pages/api/course)

Uses

CourseService
UserHelpers

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
handler	NextApiRequest, NextApiResponse	N/A	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

handler(req, res):

- Implements OpenAPI spec defined at `https://bookbazar.me/api/swagger#/Course`

Auth HTTP Handler Module (M5)

Module

AuthHttpHandler (/pages/api/auth)

Uses

MagicLinkService
SessionService
UserHelpers

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
handler	NextApiRequest, NextApiResponse	N/A	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

handler(req, res):

- Implements OpenAPI spec defined at `https://bookbazar.me/api/swagger#/Auth`

User HTTP Handler Module (M6)

Module

UserHttpHandler (/pages/api/user)

Uses

UserService
ImageService
UserHelpers

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
handler	NextApiRequest, NextApiResponse	N/A	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

handler(req, res):

- Implements OpenAPI spec defined at `https://bookbazar.me/api/swagger#/User`

User Service Module (M7)

Module

UserService (/lib/services/user.ts)

Uses

PrismaClient
ImageService

Syntax

Exported Constants

N/A

Exported Types

ModifiableUser = tuple of (name : string, imageUrl : string)
UserWithPosts = User ∪ tuple of (posts : set of Post)

Exported Access Programs

Routine name	In	Out	Exceptions
findOrCreateUser	string	User	N/A
getUserWithPosts	string	UserWithPosts	N/A
updateUser	string, ModifiableUser	User	N/A
deleteUser	string	User	N/A

Semantics

State Variables

N/A

Assumptions

- The exported access programs are only used if the calling user is authorized to use them
- Inputs to exported access programs have been validated by the caller

Access Routine Semantics

findOrCreateUser(email):

- Finds an existing user in the database by email or creates ones if none is found

getUserWithPosts(id):

- Gets a user from the database by ID and populates the user with their posts

updateUser(id, updatedUser):

- Updates a user in the database by ID using the given object

deleteUser(id):

- Deletes a user from the database by ID
- Deletes user image using ImageService

Course Service Module (M8)

Module

CourseService (/lib/services/user.ts)

Uses

PrismaClient

Syntax

Exported Constants

N/A

Exported Types

CourseWithBooks = Course ∪ tuple of (books : seq of Book)
CourseWithDept = Course ∪ tuple of (dept : Dept)

Exported Access Programs

Routine name	In	Out	Exceptions
getCourseWithBooks	string	CourseWithBooks	N/A
getPostsForCourse	string, integer, integer, boolean	seq of Post ∨ seq of PostWithUser	N/A

Semantics

State Variables

N/A

Assumptions

- The exported access programs are only used if the calling user is authorized to use them
- Inputs to exported access programs have been validated by the caller
- The number of books for a course will not grow to an unmanageable amount (e.g. > 10)

Access Routine Semantics

getCourseWithBooks(id):

- Gets a course by ID and populates the books for that course

getPostsForCourse(id, length, page, includeUser):

- Get posts for a course ID
- The number of posts to return is determined by length
- The offset of posts to return is determined by page
- The user in posts is only populated if includeUser is true

Post Service Module (M9)

Module

PostService (/lib/services/post.ts)

Uses

PrismaClient
ImageService

Syntax

Exported Constants

N/A

Exported Types

PostWithUser = Post $\cup$ tuple of (user : User)
CreatablePost = tuple of (price : integer, description : string, imageUrl : string, bookId : string)
UpdatablePost = tuple of (price : integer, description : string, imageUrl : string, status : PostStatus)

Exported Access Programs

Routine name	In	Out	Exceptions
getPost	string, boolean	Post $\vee$ PostWithUser	N/A
createPost	CreatablePost	Post	N/A
updatePost	string, UpdatablePost	Post	N/A
deletePost	string	N/A	N/A

Semantics

State Variables

N/A

Assumptions

- The exported access programs are only used if the calling user is authorized to use them
- Inputs to exported access programs have been validated by the caller

Access Routine Semantics

getPost(id, includeUser):

- Gets a post from the database by ID and populates the post with the corresponding user
- The user is only populated if includeUser is true

createPost(post):

- Create a new post in the database

updatePost(id, post):

- Updates a post in the database

deletePost(id):

- Deletes a post from the database
- Deletes post image using ImageService

Book Service Module (M10)

Module

BookService (/lib/services/book.ts)

Uses

PrismaClient
GoogleBooksSearch

Syntax

Exported Constants

N/A

Exported Types

PopulatedBook = Book ∪ tuple of (posts : set of Post ∨ set of PostWithUser, courses : set of CourseWithDept, googleBook : tuple of (* : *))

Exported Access Programs

Routine name	In	Out	Exceptions
getPopulatedBook	string, boolean, integer, integer	PopulatedBook	N/A

Semantics

State Variables

N/A

Assumptions

- The exported access programs are only used if the calling user is authorized to use them
- Inputs to exported access programs have been validated by the caller

Access Routine Semantics

getPopulatedBook(isbn, includeUsers, postsLength, postsPage):

- Gets a book from the database by ISBN and populates its posts, courses, and Google books data (using GoogleBooksSearch)
- The number of posts to return is determined by postsLength
- The offset of posts to return is determined by postsPage
- The user tuples inside posts are only populated if includeUsers is true

Session Service Module (M11)

Module

SessionService (/lib/services/session.ts)

Uses

PrismaClient
Tokens

Syntax

Exported Constants

N/A

Exported Types

SessionWithUser = Session $\cup$ tuple of (user : User)

Exported Access Programs

Routine name	In	Out	Exceptions
getSessionWithUser	string	SessionWithUser $\vee$ null	N/A
createSession	string, string	Session	N/A
deleteSession	string		N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

getSessionWithUser(sessionToken):

- Finds session with hashed version of passed token
- If no session exists, return null
- If session is expired, delete it from database and return null
- If session will expire within a month, refresh token expiration date
- Return most recent version of the session

createSession(sessionToken, userId):

- Creates a session in the database using a hashed version of the passed tokenand an expiration date of six months from the current time for the passed user ID

deleteSession(sessionToken):

- Deletes session matching passed token from database

Magic Link Service Module (M12)

Module

MagicLinkService (/lib/services/magic.ts)

Uses

PrismaClient
SgMail
UserService
SessionService
Tokens
Environment

Syntax

Exported Constants

N/A

Exported Types

TokenWithExpiration = tuple of (token : string, expirationDate : Date)

Exported Access Programs

Routine name	In	Out	Exceptions
sendMagicLink	string		N/A
consumeMagicLink	string	TokenWithExpiration ∨ null	N/A

Semantics

State Variables

N/A

Assumptions

- All passed emails are valid McMaster emails

Access Routine Semantics

sendMagicLink(email):

- Creates a new verification email database entry that expires one hour from the current time
- Sends a email with a login link to `/magic?token=VERIFICATION_TOKEN` to the specified email

consumeMagicLink(verificationToken):

- Deletes new verification email database entry (so that it is only used once)
- If verification email has not expired, return null
- Find existing user with email or create one if it does not exist using UserService
- Create a new session using SessionService
- Return created session and related un-hashed token

Image Service Module (M13)

Module

ImageService (/lib/services/image.ts)

Uses

AwsSdk
Environment

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
upload	Blob	string	N/A
delete	string	N/A	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

upload(file):

- Format image so that it can be stored
- Upload image to cloud storage
- return a publicly-accessible permalink to the uploaded image

delete(url):

- Delete image from cloud storage

Campus Store Scrapper Module (M14)

Module

CampusStoreScrapper (/lib/campus-store-scrapper.ts)

Uses

Algolia
GoogleBooksSearch
PrismaClient

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
indexTextbooks	N/A	N/A	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

indexTextbooks():

- To be run on a schedule once per week
- Get all textbooks from campus book store
- Get Google books ID for each textbook using GoogleBooksSearch
- Update or create found book, course, and department entries in database
- Update Algolia index to match database

Prisma Client Module (M15)

Module

PrismaClient

Uses

N/A

Syntax & Semantics

`https://www.prisma.io/docs/reference/api-reference/prisma-client-reference`

AWS SDK Module (M16)

Module

AwsSdk

Uses

N/A

Syntax & Semantics

`https://docs.aws.amazon.com/AWSJavaScriptSDK/latest/index.html`

Sendgrid Mail Module (M17)

Module

SgMail

Uses

N/A

Syntax & Semantics

`https://docs.sendgrid.com/api-reference`

Algolia Search Module (M18)

Module

Algolia

Uses

N/A

Syntax & Semantics

`https://www.algolia.com/doc/api-client/getting-started/what-is-the-api-client/javascript`

Google Books Search Module (M19)

Module

GoogleBooksSearch

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
getGoogleBooksData	string	GoogleBook ∨ null	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

getGoogleBooksData(isbn):

- Gets Google Books data using a book's ISBN
- Makes a call to the Google Books API
- Returns a GoogleBooks object if the response contains relevant data
- Returns null if the response does not contain relevant data

Tokens Module (M20)

Module

Tokens (/lib/helpers/backend/tokens.ts)

Uses

N/A

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
createToken	N/A	string	N/A
hashToken	string	string	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

createToken():

- Creates a cryptographically random universally unique identifier

hashToken(token):

- Irreversibly hashes a token

Environment Module (M21)

Module

Environment (/lib/helpers/backend/env.ts)

Uses

N/A

Syntax

Exported Constants

BASE_URL: string – URL that Book Bazar is currently hosted at (e.g. bookbazar.me)
AWS_API_KEY: string – API key used to connect to AWS services
SENDGRID_API_KEY: string – API key used to connect to the Sendgrid service
SENDGRID_EMAIL_FROM: string – Email address to send Sendgrid emails from
IS_E2E: boolean – Whether Book Bazar is currently being run in end-to-end testing mode

Exported Types

N/A

Exported Access Programs

N/A

Semantics

State Variables

N/A

Assumptions

- An exception will be thrown if any exported constant cannot be initialized to a valid value

Session Cookie Module (M22)

Module

SessionCookie (/lib/helpers/backend/session-cookie.ts)

Uses

Environment

Syntax

Exported Constants

SESSION_TOKEN_COOKIE: string – Key of cookie used to store session token

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
createSessionCookie	string, Date	string	N/A
createDeleteSessionCookie	N/A	string	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

createSessionCookie(sessionToken, expirationDate):

- Creates a cookie at SESSION_TOKEN_COOKIE that contains the passed sessionToken and expires at expirationDate

createDeleteSessionCookie():

- Creates a cookie that, when set, deletes the session token cookie from a users browser

User Helpers Module (M23)

Module

UserHelpers (/lib/helpers/backend/user-helpers.ts)

Uses

UserService
SessionService
SessionCookie

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
getCurrentUser	NextApiRequest, NextApiResponse	User $\vee$ null	N/A
isAuthenticated	NextApiRequest, NextApiResponse	boolean	N/A
isUserProfileCompleted	User	boolean	N/A

Semantics

State Variables

N/A

Assumptions

N/A

Access Routine Semantics

getCurrentUser(req, res):

- Gets and returns currently logged in user
- Refreshes session token cookie with latest value

isAuthenticated(req, res):

- Returns true if getCurrentUser returns a user tuple and false if null is returned

isUserProfileCompleted(user):

- Returns true if user name has a value

5 Database Design

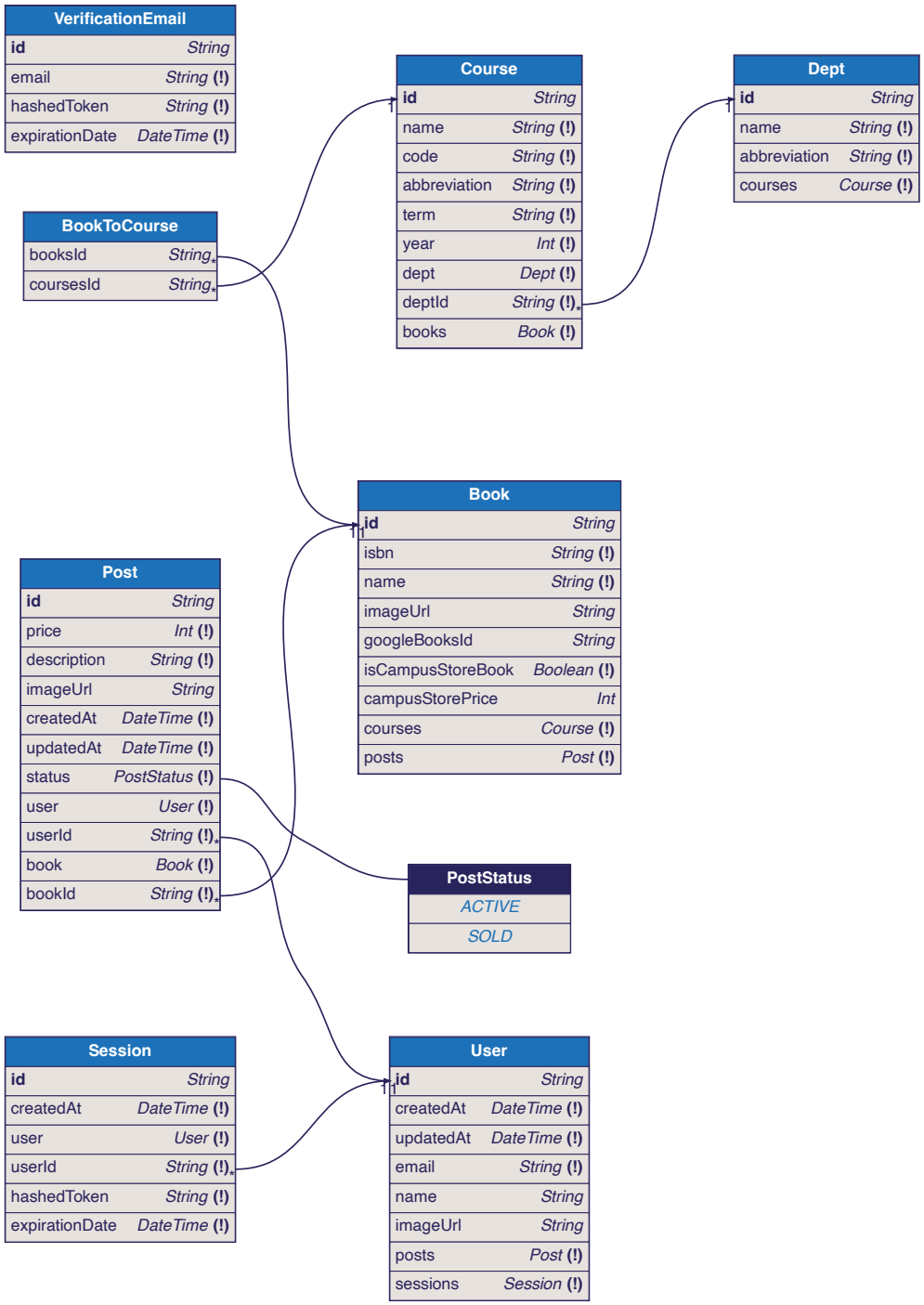

Figure 3: Database Design