CS 319 - Bilcon Web Application D4

Section 1 - Team 2

Group Members:

Öykü Demir - 22101934

Begüm Kunaç - 2210383

Hakan Muluk - 22102512

Gün Taştan - 22101850

Mehmet Onur Özdemir - 22102293

Mustafa Gökalp Gökdoğan - 22102936

Table of Contents

Table of Contents 1
1. Design Goals 2
1.1 Maintainability 2
1.2 Usability 2
1.3 Design Tradeoffs 3
1.3.1 Maintainability vs. Performance: 3
1.3.2 Functionality vs. Simplicity: 3
1.3.3 Flexibility vs. Specificity: 3
2. High Level Software Architecture 4
2.1 Overview 4
2.2 Subsystem Decomposition 4
2.2.1 User Interface Layer 5
2.2.2 Application Layer 6
2.2.3 Database Interface Layer 7
2.2.4 Data Layer 8
2.3. Deployment Diagram 9
2.4. Hardware/Software Mapping 10
2.5. Persistent Data Management 10
2.6 Access Control and Security 11
2.7 Boundary Conditions 14
2.7.1 Initialization 14
2.7.2 Termination 14
2.7.3 Failure 15

1. Design Goals

1.1 Maintainability

The system's functionalities will be designed such that the system structure will be maintainable even if there exists new demand from the users for additional functionalities. The type of additional functionalities that the users will be demanding will be thought of before deployment, and the design will be done accordingly. Also, the changing and/or future possibility of technology changes of the system will be considered, making the system maintainable. The system will essentially support change and editorial prospects of implementation, making the system overall maintainable during time of deployment.

We are dedicated to applying best practices for improved maintainability in our project. We will divide the system into discrete parts with clearly defined interfaces, each in charge of handling particular functions, by using a modular design approach. Because of its modular design, the system is easier to understand, update, and replace, and can adapt to changing needs. Thorough documentation, encompassing system architecture, design choices, and codebase specifics, will serve as a fundamental component and offer priceless assets for present and future developers. We will be able to track changes, roll back when needed, and preserve a stable baseline by using release tagging with version control systems such as Git. In order to promote code quality and early issue identification, team members will be aligned with coding standards and best practices through weekly code reviews. Together, these techniques help to create a robust system that is ready for future improvements and seamless maintenance in addition to its efficient operation.

1.2 Usability

There exist some common needs for all Bilkent students such as lecture books, finding an item that was lost etc. Bilcon is a web based application that is for meeting these requirements of students in a single platform that combines solutions for different types of requirements. Bilcon supplies its functionalities in a user-friendly manner, by separating different roles (such as customer, seller) into different screens. Also, registering or logging in to the system does not need the users to do complex tasks, they can register or log in by their bilkent mails. Lastly, the separated screens are simple, easy to learn and consist of self-explanatory components; for example, the action(s) that a button does is clearly written on it.

The conventional practices of common knowledge applications will essentially be replicated so that the users will be able replicate their exante behavior of usage from analogous applications.

1.3 Design Tradeoffs

1.3.1 Maintainability vs. Performance:

The emphasis on maintainability in our design may compromise performance optimization. Although comprehensive documentation and modular designs facilitate maintenance, some performance gains may have to be given up. Finding the perfect balance between maintainability and performance, given the particular needs and demands of our users, will require careful consideration.

1.3.2 Functionality vs. Simplicity:

Bilcon provides many features for its users. However, while offering a rich set of features, the user interface should still be simple and minimalistic for a better user experience. Finding the proper balance between these two is very crucial in order to offer all functionalities of the application without an overwhelming experience for users.

1.3.3 Flexibility vs. Specificity:

Bilcon has several use cases. Designing the system to be both flexible for different use cases and focused on a specific purpose can be challenging. Designing the system with multiple different functionalities may lead to complexity, while being too specific can limit creativity and the proper development of certain features of the application. Balancing these two is very significant in terms of keeping the application’s features varying but also intact.

2. High Level Software Architecture

2.1 Overview

2.2 Subsystem Decomposition

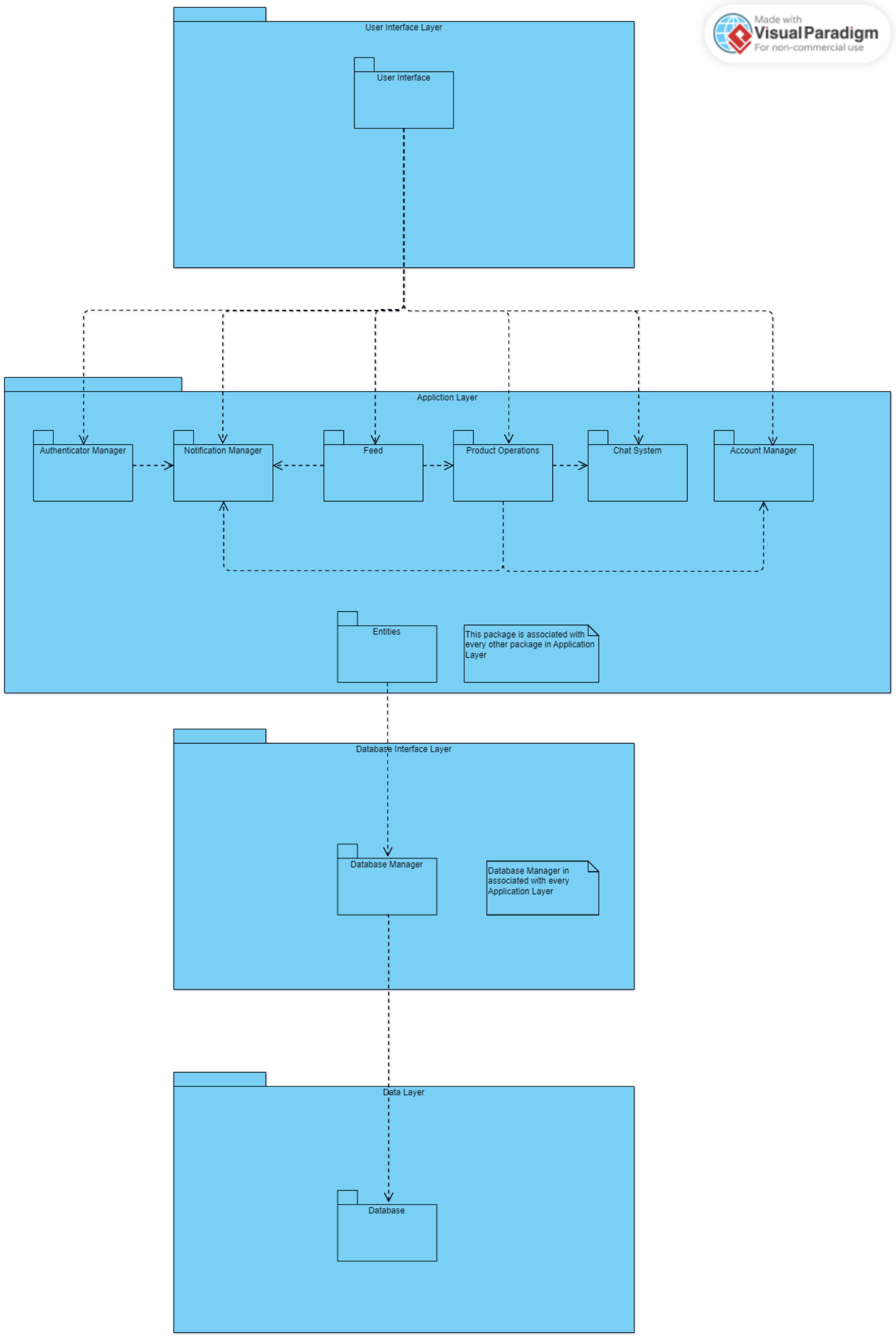


Figure 1: Subsystem Decomposition

We preferred a 4-layer structure in subsystem decomposition. These are User Interface Layer, Application Layer, Database Layer and Data Layer. The reason for dividing it this way was to simplify the complex system by dividing it into basic subsystems grouped with common features, indicating what to show to the user, what the application can do within itself (business logic), and their relationship with the product and user data.

2.2.1 User Interface Layer

User Interface layer manages the interactions between users and different screens (displays, buttons, menus and dropdowns). These screens allow users to navigate between different tabs in the application; it allows users to see second hand market, rental transactions, lost property transactions, etc. If they want to make transactions or make observations in the application, User Interface package provide the necessary communication system with the application and send these requests to the Application layer packages, which is the logic part. This means that User Interface layer is associated with Application layer.

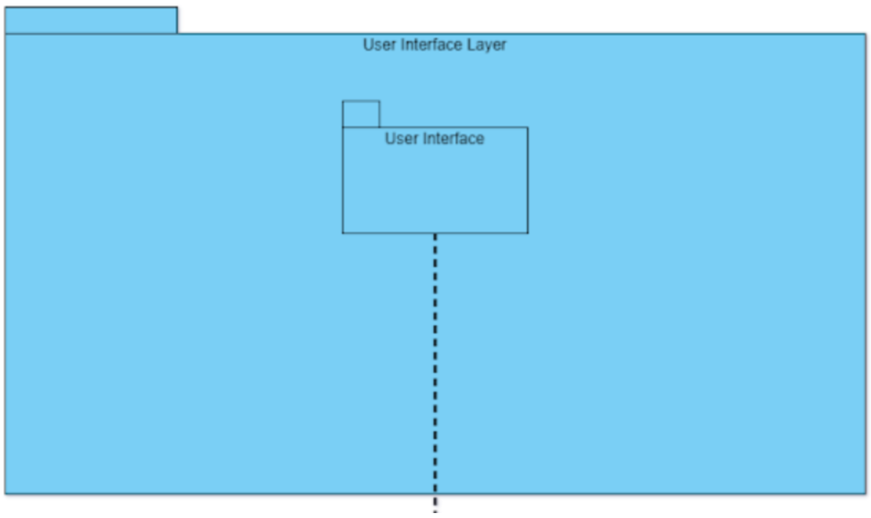


Figure 2: User Interface Layer

UserInterface:

- This package has screens to provide interactions between users and application. These screens include:
    - Sign up screen
    - Log in screen
    - Second hand market (listing a second hand item and inspecting product details)
    - Lost item screen (posting lost and found item, viewing listed lost items)
    - Course trading screen
    - Private lesson listing screen
    - Chat screen
    - Rate screen (rating a user and view)

- Recommendation screen

- This package has associations with Authenticator Manager, Notification Manager, Feed, Product Operations, Chat System, Account Manager Packages of Application Layer.

2.2.2 Application Layer

This layer is the business logic layer and it has packages of control objects that manipulate and manage data according to received data and user operations' results. This layer has associations with User Interface layer (where data and operation results come from) and Database Interface layer (where associated changes are going to be reflected to).

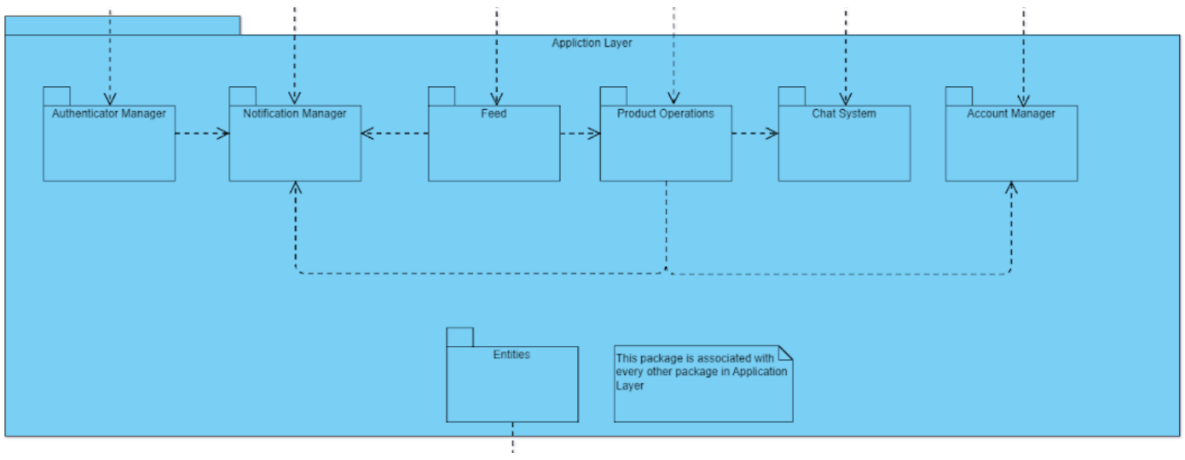


Figure 3: Application Layer

AuthenticatorManager:

- This package has control classes to manage user authentication which primarily ensures that the user is really the owner of the account with the given email. Also it plays a role in every login attempt that user performs.
- It has an association with UserInterface, NotificationManager packages.

NotificationManager:

- This package has control classes that manage notifications.
- It has associations with UserInterface, Feed, AuthenticatorManager, ProductOperations packages.

Feed:

- This package has control classes that manage feed (market listings, recommendation, etc.)

- It has associations with UserInterface, ProductOperations, NotificationManager packages.

ProductOperations:
- ProductOperations package include control classes that manage creating second hand item listing, lost/found product listings, rental product listing, private lesson listing, buying and renting products.
- It has associations with UserInterface, Feed, ChatSystem, NotificationManager, AccountManager packages.

ChatSystem:
- This package is responsible for managing communications between users for buying, selling, renting, borrowing, lost-found item reasons.
- It has associations with UserInterface and ProductOperations packages.

AccountManager:
- This package is responsible for account related operations such as updating user information, updating profile according to posted ratings, and products operations.
- It has associations with UserInterface and ProductOperations packages.

Entities:
- This package is responsible for holding and sending entity objects which are user created, manipulated or updated/created according to operations users have performed.
- It has an association with DatabaseInterface package.

2.2.3 Database Interface Layer

This layer has a relation and association with Application layer and Data layer. Meaning that this layer is responsible for reflecting and linking the effects of logics and operations done by Application layer to Data layer which holds required information about the system, users and products.

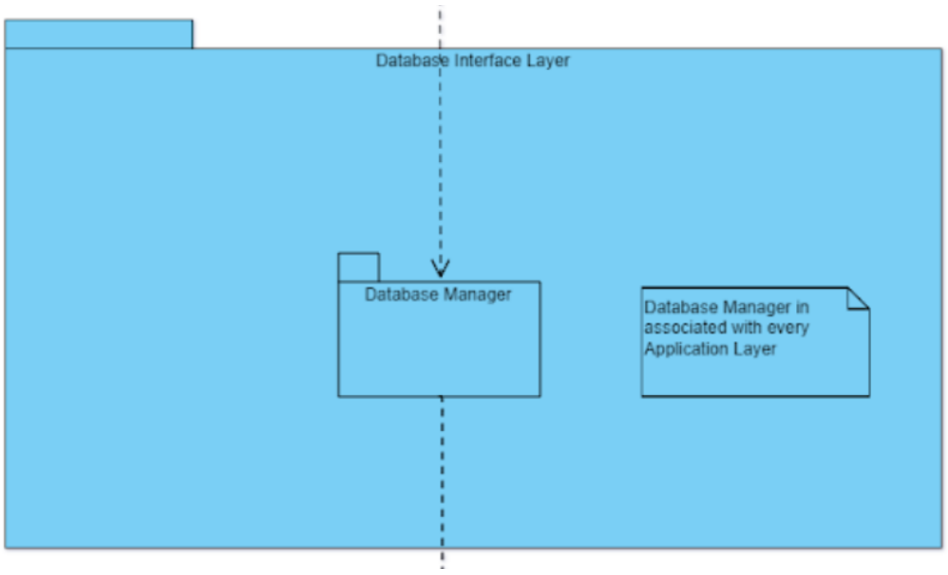


Figure 4: Database Interface Layer

DatabaseManager:

- DatabaseManager package has classes that manage database.
- This package has a relation and association with Entity Package of Application Layer and Database Package of Data Layer.

2.2.4 Data Layer

Data Layer has Database Package as its single package. This layer is associated with Database Interface Layer.

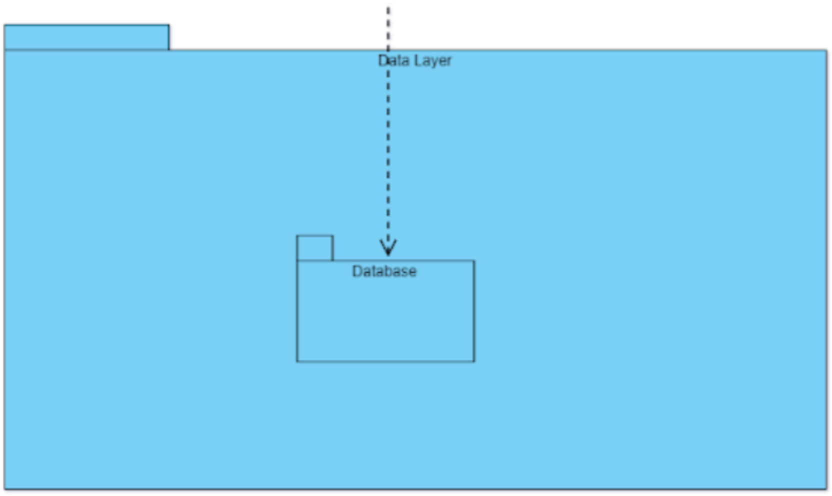


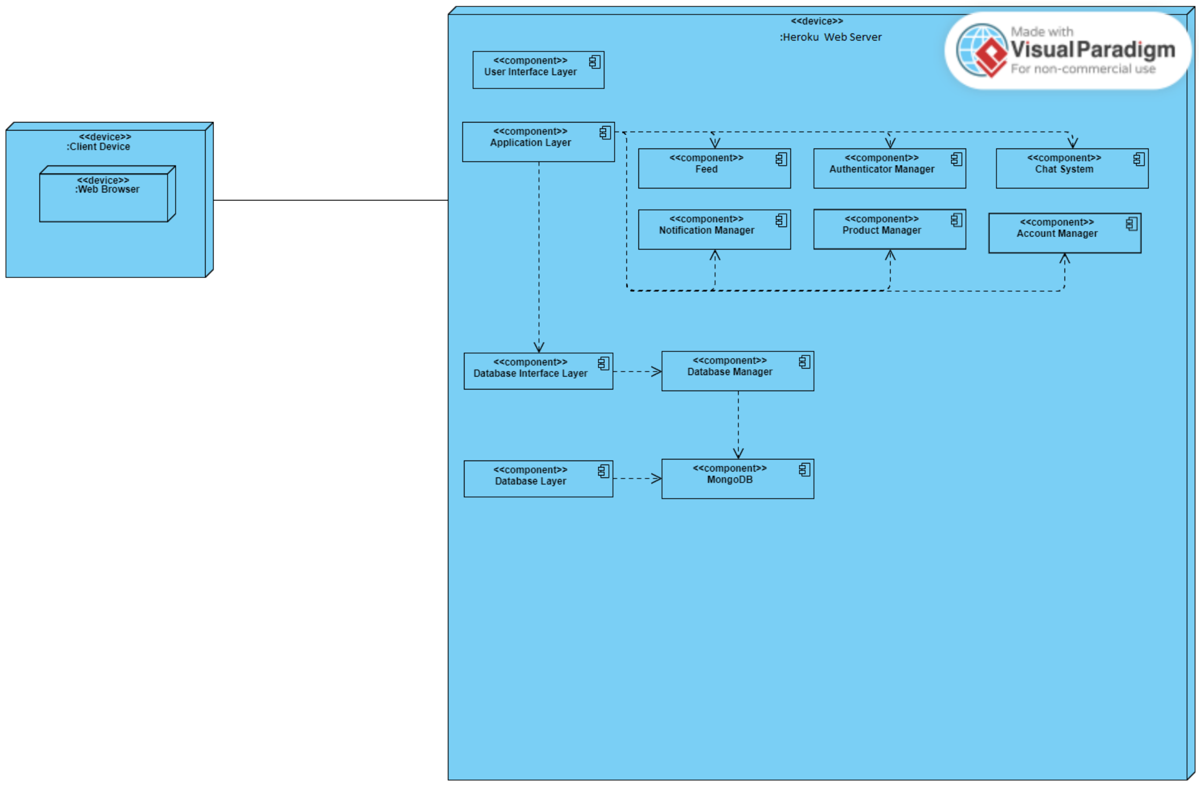
Figure 5: Data Layer

Database:

- Database is responsible for storing, securing and being able to provide operations on the data about system, products and users.
- This package is associated with DatabaseManager Package.

2.3. Deployment Diagram

The communication between the client and server is depicted in this diagram. Its layout is extremely close to that of subsystem breakdown. Classifications that function closely together and may be categorized by kind are called components, while physical aspects in the physical world are represented by objects.



<<device>>
:Heroku Web Server
Made with Visual Paradigm For non-commercial use
<<component>> User Interface Layer
<<component>> Application Layer
<<component>> Feed
<<component>> Authenticator Manager
<<component>> Chat System
<<component>> Notification Manager
<<component>> Product Manager
<<component>> Account Manager
<<component>> Database Interface Layer
<<component>> Database Manager
<<component>> Database Layer
<<component>> MongoDB
<<device>>
:Client Device
<<device>>
:Web Browser

Figure 6: Deployment Diagram

2.4. Hardware/Software Mapping

The client-side of the application is built using React.js framework. React is chosen because it is a well-structured framework for building web applications with flexibility. It also provides a proper amount of reusability. Additionally, it has sufficient documentation, which makes it easier to develop complex systems by making use of proper resources. Bilcon does not necessarily require a complex hardware system. A stable version of a browser is adequate for the website to work properly.

Express.js is used as the web application framework for Node.js. Express.js is chosen due to its easiness and flexibility in designing RESTful APIs.

MongoDB is the database of the application. Its document model is schema-less, which is considered helpful in implementing object-oriented programming methods, especially because of its flexibility.

Heroku is the main cloud based deployment platform for hosting Bilcon. Backend and frontend will be deployed on Heroku, and database deployment will be performed with MongoDB Atlas. MongoDB Atlas is used as a cloud-based database service for MongoDB, providing a secure environment for the database.

2.5. Persistent Data Management

MongoDB is a NoSQL language, and it provides a document-oriented database. This data model makes it efficient in handling dynamic data structures in the application, making the data handling process more flexible and adaptable. Document structure that is used in MongoDB creates a mapping between different objects in the application. Also since MongoDB supports arrays, it makes the handling of relationships within documents remarkably easier. In the database, objects such as users, products, and notifications will be sufficiently stored.

MongoDB Atlas, is the cloud-based database service that is used in Bilcon within MongoDB. MongoDB is very useful in database deployment and management as it can be easily integrated to other cloud-based providers, in our case, Heroku.

MongoDB and MongoDB Atlas provides a scalable and high-performance implementation of data management. It integrates with the dynamic structure of the application and MongoDB’s structure of documents help in flexible data storage, aligning with the principles of object-oriented programming.

2.6 Access Control and Security

Bilcon is a web app that cares for security and limits the users' actions via an access control system. The access control system assigns some pre-specified roles to each user type and thus, prevents possible chaos in the system. The user types that have roles are: Seller, Customer, LoserOfItem, FinderOfItem, PrivateLessonTaker, PrivateLessonGiver, CourseTrader and BilkentAuthenticator.

At the client side, users cannot access all properties of the system, but they can access the properties of their roles. For example, a customer(user in customer page) is not allowed to post an item for sale/rent. He/she should pass to the seller role by going to the seller screen in order to do that. The web-app supplies this behavior by tracking the interactions between the users and the system: after logging in, if the user has pressed to go to the customer page, he/she has a customer role. If he/she wants to transform to another role, he/she has to go to that role's screen by interacting with the system. This interaction is tracked by the system and the role of the user will be changed according to type of interaction.

At the server side, the information of each user is recorded seperately. So, if a user changes his/her data, only the data of that user is changed. For example, when a seller removes a post, nothing happens to other seller's posts. Also, a user that has a specific role cannot use the functionalities of other roles, this is supplied by both client and server sides. Finally, some critical information is encrypted before being stored in the database.

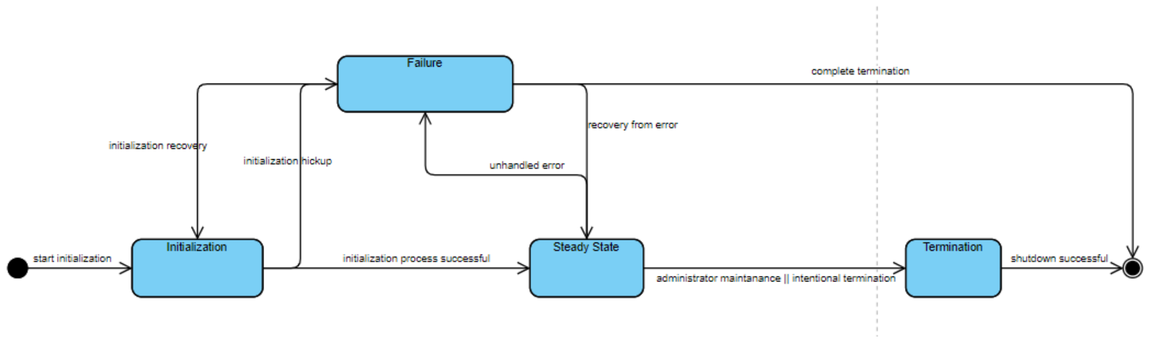
Access Control Table

	Seller	**Customer**	**Loser OfItem**	**Finder OfItem**	**Private Lesson Taker**	**PrivateLesson Giver**	**Course Trader**	**BilkentAuthenticator**
Register	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
Login	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
Change Password	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
See Lost Item Posts			**X**					
See Private						**X**		

Lesson Offers								
See Course Trade Offers							**X**	
Post Item	**X**							
Post Found Item				**X**				
Post course for trade							**X**	
See Item		**X**						
Post private lesson						**X**		
Make request of private lesson					**X**			
FilterSearchResults		**X**			**X**		**X**	
Make request of course for trade							**X**	
Search for Found Item			**X**					
Inspect Other User's Profile	**X**	**X**	**X**	**X**	**X**	**X**	**X**	

Inspect Own Profile	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
Check Received Message s	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
Send Verificati on Mail								**X**
Sign User Up								**X**
DM Another User	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
Rate other person	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
Edit Own Profile	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
Edit Post or Request	**X**		**X**	**X**	**X**	**X**	**X**	
Remove Post or Request	**X**		**X**	**X**	**X**	**X**	**X**	
Rate other	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
Create Lost Item Post			**X**					

2.7 Boundary Conditions



2.7.1 Initialization

Bilcon is inherently a WEB application, the user enters the necessary credentials and logs in to the system which starts the initialization process. The entered credentials are then used to fetch the necessary information from the database for which belongs to the user in question. The users who do not have an account can not start the initialization process, they first must register to the system, which also is a valid initialization scenario. When the system is initializing, different lost item posts, found item posts, private lesson offers, course trade offers, items that are for sale, received messages of users etc. are fetched from the database. Additionally, the maintainer of the system must initialize the running server on Heroku. Evidently, the database should also be initialized and the IP of the database should be provided to the backend of the system to ensure correct communication.

2.7.2 Termination

Bilcon can be terminated because of different reasons, which can be classifies as planned and unplanned. Firstly, when a user logs out the app, the app will be terminated, this is a planned termination. Another planned termination might occur if there is an issue about the system. Then, the admin might terminate the app in order to solve the issue. Lastly, a failure due to some exception might occur in the system and this might cause the system to terminate and this type of termination is an unplanned termination.

2.7.3 Failure

If a failure occurs, the system is restarted. If restarting does not solve the problem, the system is shut down and the developers are notified. No user can use the system when it is closed, they will be notified by an email when the system is working again.