SYSTEM ANALYSIS

Users

STATIC BON

|  |  |  |  |
| --- | --- | --- | --- |
| SYSTEM | *WOODLAND\_RECORDS\_SYSTEM* | | Part: 1/1 |
| PURPOSE System that deals with adding and editing records in the system. | | INDEXING  author: Nishedh Karki  keywords: records | |
| Cluster | Description | | |
| *USERS* | *All the users and their types.* | | |
| *SERVICES* | *Includes the courses, modules, and other services.* | | |
| *GENERAL\_USES* | *General uses include diary management and announcement management.* | | |

**Cluster Charts**

|  |  |  |  |
| --- | --- | --- | --- |
| CLUSTER | *User* | | Part: 1/1 |
| PURPOSE To handle all the users directly related activities. | | INDEXING  author: Nishedh Karki  keywords: teacher, student, user | |
| **Class/(Cluster)** | Description | | |
| *User* | *Information of all the users. Includes general information of person and their role in this system.* | | |
| *Teacher* | *Holds information of teacher related information. (Includes module and personal student)* | | |
| *Student* | *Holds information about student(course).* | | |
|  |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLUSTER | *SERVICES* | | Part: 1/1 |
| PURPOSE To handle all the services in the system. | | INDEXING  author: Nishedh Karki  keywords: logs , services | |
| **Class/(Cluster)** | Description | | |
| *Course* | *Holds details of courses in the system.* | | |
| *Module* | *Holds details of modules and what course it belongs to.* | | |
| *PersonalTutortRequests* | *Holds the information of personal tutor requests made by student to a teacher.* | | |
| *Assignments* | *Holds assignments assigned for a module* | | |
| *Submissions* | *Holds submission data by student of assignment* | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLUSTER | *GENERAL\_USES* | | Part: 1/1 |
| PURPOSE To handle all general uses in the system. | | INDEXING  author: Nishedh Karki  keywords: logs , services | |
| **Class/(Cluster)** | Description | | |
| *Log* | *Holds all the personal logs by all users.* | | |
| *Announcement* | *Includes all the announcements made by admin.* | | |
|  |  | | |

Class Charts:

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *USER* | | Part: 1/1 |
| TYPE OF OBJECT  A user registered in the system. | | INDEXING  cluster:  USERS | |
| Queries | Name, dob, address, phone number, email, password, role, uID. | | |
| Commands | Add, update, delete, parseUser. | | |
| Constraints | uID is unique to every user. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *Teacher* | | Part: 1/1 |
| TYPE OF OBJECT  Teacher and admins in system. | | INDEXING  cluster:  USERS | |
| Inherits from | User | | |
| **Queries** | Module, salary, role. | | |
| Commands | Add personal Student, get personal student. | | |
| Constraints | Admin can’t have a module whereas if role is teacher then module, is a must. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *Students* | | Part: 1/1 |
| TYPE OF OBJECT  Students in the system. | | INDEXING  cluster:  USERS | |
| Inherits from | User | | |
| **Queries** | Course | | |
| Commands | Get personal teacher. | | |
| Constraints | Course is a must. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *Course* | | Part: 1/1 |
| TYPE OF OBJECT  A course being taught in the university. | | INDEXING  cluster:  SERVICES | |
| Queries | Name, duration, description | | |
| Commands | Add, delete, add Module. | | |
| Constraints | Name must be unique. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *Module* | | Part: 1/1 |
| TYPE OF OBJECT  A module in a Course. | | INDEXING  cluster:  SERVICES | |
| Queries | Name, credit, duration, description. | | |
| Commands | Add, update, delete. | | |
| Constraints |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *PersonalTutorRequests* | | Part: 1/1 |
| TYPE OF OBJECT  A request for personal tutor. | | INDEXING  cluster:  SERVICES | |
| Queries | Student, teacher, module | | |
| Commands | Find, Add, update, delete, parse. | | |
| Constraints | Student cannot have multiple requests for same module. | | |

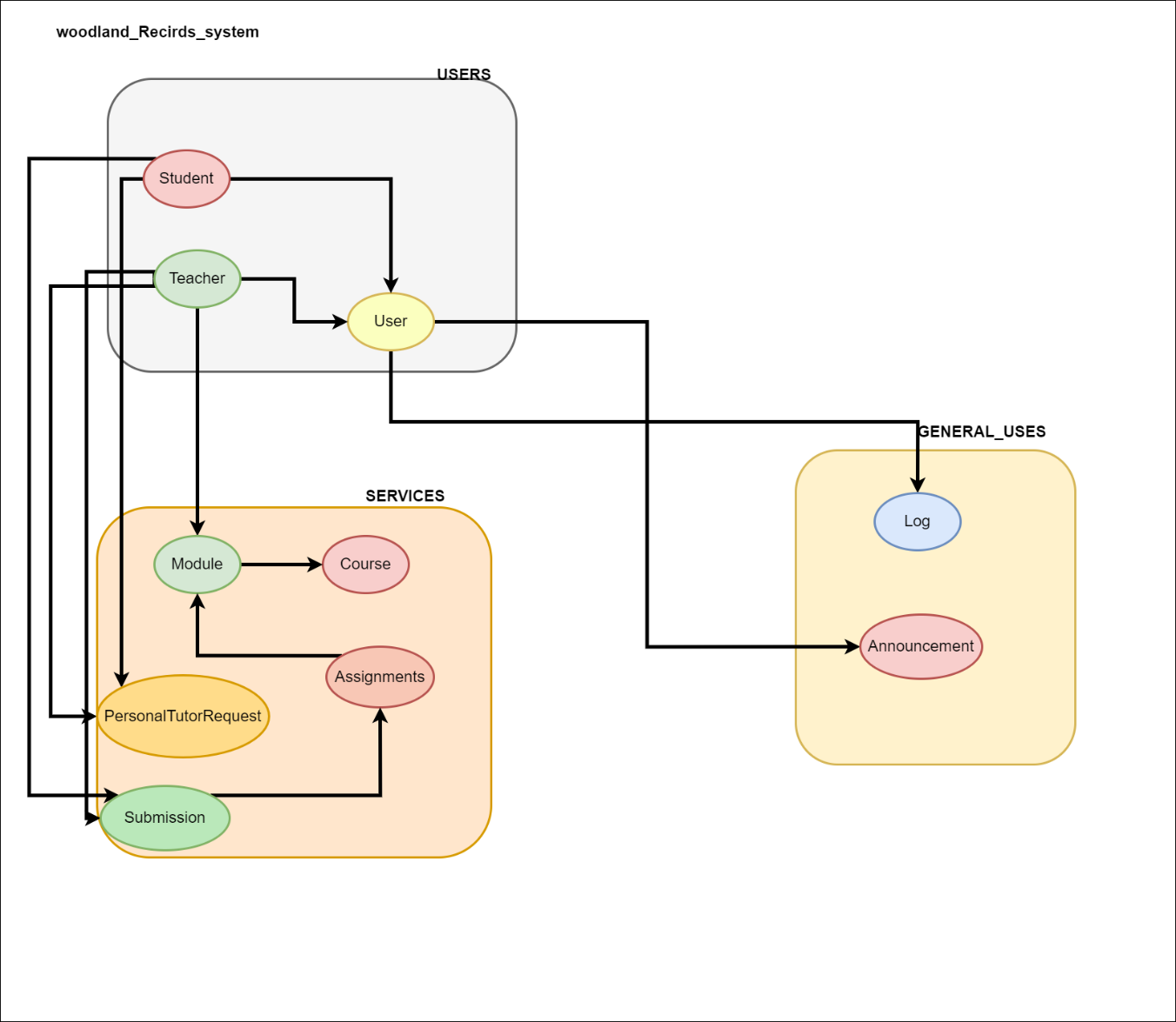
|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *Assignment* | | Part: 1/1 |
| TYPE OF OBJECT  An assignment given by teacher in certain module. | | INDEXING  cluster:  SERVICES | |
| Queries | Title, date, due, extension, module | | |
| Commands | Find, Add, update, delete. | | |
| Constraints |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *Submisson* | | Part: 1/1 |
| TYPE OF OBJECT  A submission from student on a assignment. | | INDEXING  cluster:  SERVICES | |
| Queries | Student, date, grade, teacher, filename, comment, assignment, feedback | | |
| Commands | Find, findforStudent, add, delete, grade. | | |
| Constraints | A new submission on same assignment overwrites the previous submission | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *Log* | | Part: 1/1 |
| TYPE OF OBJECT  log is a personal log (diary) of a user. | | INDEXING  cluster:  SERVICES | |
| Queries | User, text. date | | |
| Commands | Find, add, edit, delete | | |
| Constraints | - | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CLASS | *Announcement* | | Part: 1/1 |
| TYPE OF OBJECT  Announcement made by admin for all user | | INDEXING  cluster:  SERVICES | |
| Queries | User, text, title, date | | |
| Commands | Find, add, delete | | |
| Constraints | - | | |

Architecture diagram



Dynamic Bon

EVENTS CHARTS:

|  |  |  |  |
| --- | --- | --- | --- |
| EVENTS | *ADMIN\_RECORDS\_MANAGEMENT\_EVENTS* | | Part: 1/1 |
| **COMMENT**  System events that will occur in records management system. | | INDEXING:  Created: 06/02/2022 | |
| External Events | **Involved Object Types** | | |
| Request to add Student | USER, STUDENT, COURSE | | |
| Request to add Teacher | USER, TEACHER, MODULE | | |
| Request to add a course | COURSE | | |
| **Request to add module** | COURSE, MODULE | | |
| Request to add Announcements | ANNOUNCEMENT, USER | | |
| **Request to edit Teacher** | USER, TEACHER, MODULE | | |
| Request to edit Student | USER, STUDENT, COURSE | | |
| **Request to edit a Module** | MODULE, COURSE | | |
| **Request to delete Teacher** | USER, TEACHER, MODULE | | |
| **Request to delete Student** | USER, STUDENT, COURSE | | |
| **Request to delete a Module** | MODULE, COURSE | | |
| **Request to delete a course** | COURSE | | |
| Request to delete Announcements | ANNOUNCEMENT, USER | | |
| Request to add Log | LOG, USER | | |
| Request to delete Log | LOG, USER | | |
|  |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| EVENTS | *TEACHER\_SIDE\_INFORMATION\_PORTAL\_EVENTS* | | Part: 1/1 |
| **COMMENT**  System events that will occur in Student portal from teachers side. | | INDEXING:  Created: 06/02/2022 | |
| External Events | **Involved Object Types** | | |
| Request to add assignments | USER, TEACHER, ASSIGNMENT, MODULE | | |
| Request to grade a submission | USER, TEACHER, ASSIGNMENT, MODULE, SUBMISSION, STUDENT | | |
| Request to delete assignment | USER, TEACHER, ASSIGNMENT, MODULE | | |
| **Request to delete personal tutor request** | PERSONAL\_TUTOR\_REQUESTS, STUDENT, TEACHER, MODULE | | |
| Request to accept a personal tutor request | PERSONAL\_TUTOR\_REQUESTS, STUDENT, TEACHER, MODULE | | |
| Request to terminate personal tutor request | PERSONAL\_TUTOR\_REQUESTS, STUDENT, TEACHER, MODULE | | |
|  |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| EVENTS | *STUDENT\_SIDE\_INFORMATION\_PORTAL\_EVENTS* | | Part: 1/1 |
| **COMMENT**  System events that will occur in Student portal from students’ side. | | INDEXING:  Created: 06/02/2022 | |
| External Events | **Involved Object Types** | | |
| Request to add Submission | USER, ASSIGNMENT, SUBMISSION, STUDENT | | |
| **Request to create personal tutor request** | PERSONAL\_TUTOR\_REQUESTS, STUDENT, TEACHER, MODULE | | |
|  |  | | |

OBJECT CREATION CHARTs

In user cluster

|  |  |  |  |
| --- | --- | --- | --- |
| CREATION | *RECORDS\_MANAGEMENT\_SYSTEM* | | Part: 1/1 |
| **COMMENT**  All the classes that create objects in the in User cluster. | | INDEXING:  Created: 06/02/2022 | |
| Class | **Creates instances of** | | |
| STUDENT | USER, COURSE, MODULE | | |
| **TEACHER** | USER, COURSE, MODULE | | |
|  |  | | |

In Services Cluster

|  |  |  |  |
| --- | --- | --- | --- |
| CREATION | *RECORDS\_MANAGEMENT\_SYSTEM* | | Part: 1/1 |
| **COMMENT**  All the classes that create objects in the in Services cluster. | | INDEXING:  Created: 06/02/2022 | |
| Class | **Creates instances of** | | |
| *Module* | COURSE | | |
| *PersonalTutortRequests* | MODULE, TEACHER, STUDENT | | |
| *TEACHER* | *USER* | | |
| *STUDENT* | *USER* | | |
| *Assignment* | *MODULE* | | |
| *Submission* | *ASSIGNMENT, TEACHER, STUDENT* | | |

|  |  |  |  |
| --- | --- | --- | --- |
| CREATION | *GENERAL\_USES* | | Part: 1/1 |
| **COMMENT**  All the classes that create objects in the in general uses cluster. | | INDEXING:  Created: 06/02/2022 | |
| **Class/(Cluster)** | Description | | |
| *Log* | *USER* | | |
| *Announcement* | *USER* | | |
|  |  | | |

SYSTEM SCENARIO CHARTS: -

Listing all the existing system usages is impractical and not very useful. Hence, simple actions are removed and complex ones are Listed below.

|  |  |  |  |
| --- | --- | --- | --- |
| SCENARIOS | *WOODLAND\_SYSTEM* | | Part: 1/1 |
| **COMMENT**  All the classes that create objects in the in general uses cluster. | | INDEXING:  Created: 06/02/2022 | |
| ***Submit for assignment:***  *A student goes to assignments page from module, then user goes to submission page from respective assignment, user submits his submission with appropriate form.* | | | |
| ***Request for personal tutoring and accept request:***  *A student navigates to modules page and goes to request personal tutors. From there the user selects a teacher for that module for personal tutoring. A teacher goes to personal Student page and selects a request from a student to accept as personal student.* | | | |
| ***Deleting a module:***  *An admin goes to modules page and clicks delete button on desired module.* | | | |

Dynamic diagrams:-

5. System build and technical notes:

The system is built around NodeJS framework of JavaScript as server-side programming language. Express library is used to handle all the Http request made to the server. EJs template is used to create Html template for the front end to handle the data provided by the server in the client-side. CSS is used to make website better with bootstrap library to sprinkle some flavor. The system utilizes mongo DB online database to store all the data in the system. The project overall follows an MVC pattern which is one of the most popular design practices around.

Connect mongo DB URI :- mongodb+srv://bliss:2eRYfCRdRuVMXi7M@woodland.pfprl.mongodb.net/myFirstDatabase?retryWrites=true&w=majority.

Bcrypt library is used to hash the password for the user. Currently the system sets the password of the user as their own name.

Testing feature:

Additional admin feature on “/admin/test” is added to add several dummy data to the database so that redundant work of adding multiple data by hand is removed. For this go to session control files inside controller folder and set dev\_mode to true.

DEV\_MODE sets the session as admin every single time i.e., logs the user in automatically in every request for easier use. It also allows for axios request made in admin/test route as well. The test route allows the user to automatically add dummy courses, add dummy modules to the courses, add students to the courses and add teachers to the modules. This way testing and evaluation is easier with more data pseudo-users.

Text

Description automatically generated

Difficult Code Sections :-

A screenshot of a computer

Description automatically generated

The use of the provided function is to parse either teacher or student object into a plain jSon type format with no nested values. Its usefulness comes while setting the data on a table where we don’t know what kind of data is being transferred. If a data is nested in any of the situation it might break the program or have unwanted results. As such same table generating code can be utilized for both teachers and students. It was difficult to code initially since the student or teacher object returned from database is not mutable.

ON GOING TESTING METHODOLOGIES:

Black box, unit and agile testing is done mostly throughout the system. Black box testing refers to testing the system by using the system and testing by hand with no regards for how the system was built or what codes are used. Hence, both random and carefully set inputs are selected to see if the system behaves as intended and if the requirement that it is trying to achieve is achieved.

Unit testing is done by individual developer (i.e. team member working on certain functionality) while developing the system. The sole responsibility to test the system through unit testing goes to the person working on their parts. All the small parts of the system need to be working as intended before pushing it to the main branch on remote git repository as an error may break the whole system.

Agile testing includes black box testing in each remote main git push, so the bugs and errors are fixed right away to not have any errors later on.

TEST STRATEGY OVERVIEW

Testing is planned to be performed by every member after each successful push to the remote git repository. Since the database is set remote as well, we can be sure that everyone is using the same database and if a member adds a data, then all of us can use that to perform tests. This saves the members from having to add data every single time. In the beginning all the tests are black box testing then more nuances of testing is applied to see how the system would behave. However, the more nuanced tests are only carried out later in development. Automated tests are not written due to time constraints.

The focus of the test is to make sure that functionalities are performed well, check what breaks the system. The tests during the development phase would answer the following the question:

1. Is the system working?
2. Is the intended functionality working?
3. What happens if u provide garbage data?
4. What happens if u force certain behaviors (for example if u provide URL to delete an non-existent record)?
5. Does the system crash at any point?

During the beginning phase if the answer to all these questions came positive(not yes to all but positive as in everything is fine) then the system is considered working and then moved to different requirement from back log.

If an error or bug is found in the system, then it is discussed, and a time is fixed for that bug to be fixed. Generally, all bugs are fixed on same day so everyone can move on, whereas some complex bugs are fixed by team’s effort to research and provide possible solutions in meeting. This way black box testing is performed on the system.