Abstract

G-HELP is an online support system designed for the department of Computer Science, aimed at enabling new graduate students to get vital information and services ranging from academics to social events on pre-arrival and while in St. John’s. Each new student is assigned a mentor (an old willing student who is registered), and the assignment of the mentor is done by the system’s administrator. G-HELP system also offers interested host families the opportunity to register so that they may get new students to offer hosting services. In general, the G-HELP system is designed on the model of simplicity and usability, enabling ease of use and efficiency.

Table of Contents

Introduction

This project is the development of a web-based system, called G-HELP, which will provide support services/information to new Computer Science graduate students in St. John's campus. The project begins with requirements elicitation from the client and the requirements are captured as uses cases, which are also described with text and modeled with UML diagram, for clarity. An analysis of this system was done carried out using these three models, the functional model (use cases, scenarios), the object model (class and object diagram) and the dynamic model (sequence diagrams). These analysis models helps to understand better the relationships among various objects of the system and their attributes. The design goals of the system were identified from the non-functional requirements, followed by object modeling and decomposition of the system into its subsystems. The system architectural style selected for the G-HELP system is a three-tier architecture, which provides a front-view interface for users to interact with the system and the application logic which controls the manipulation of data and then the database (storage) which sits at the backend. As part of the system design, prototypes were drawn from use cases which was shown to the client before implementation of the system commenced. G-HELP was implemented using node.js and MongoDB database. Unit and functional testing were carried out at the completion of each iteration. At the end, a full system test was done to ensure that the system is consistent with the specified user requirements. It is noteworthy to state that for the system to meet deadline, the concepts of Configuration Management, Rationale Management, and Project Management were employed. These were needed for version control, justifiable development decisions, based on available alternatives, proper task and activities planning.

2.0 Requirements elicitation

The requirements of the system were collected from the client as use cases which are highlighted and then described in the following sections.

2.1 Use cases:

1. Register Mentor
2. Register Host Family
3. Assign Mentor
4. View Course Information
5. Login
6. Review Mentor
7. Add Event
8. Communication Between Students
9. List Staff
10. Create TimeLine

2.1.1 Register Mentor

This allows existing student that has met predefined criteria (semester > 2) for becoming a mentor to register themselves on the system as mentor for the new graduate students.

2.1.2 Register Host Family

This allows interested host families to provide the system with necessary information about themselves and their preferences for new students.

2.1.3 Assign Mentor

This allows the administrator to assign mentor to new graduate students.

2.1.4 View Course Information

This provides the students with necessary information on the different courses within the department.

2.1.5 Login

Provides authenticated and authorized users access to the G-Help system.

2.1.6 Review Mentor

This provides new students the option to give a review about their mentor.

2.1.7 Add Event

This allows the administrator to provide students with information about upcoming events–academic and extracurricular.

2.1.8 Communication between Students

This provides a medium for interaction among students–old and new.

2.1.9 List Staff

This provides information about departmental staff.

2.1.10 Create TimeLine

This allows the administrator to update deadline for different events or activities.

2.2 Use case diagram

2.2.1 Actors’ description

New student: graduate student that is either in semester 1 or semester 2

Old Student: graduate student in semester 3 or higher and who willingly registers to become a Mentor.

Host Family: resident in St’ John’s who is interested in having student(s) to host for social interaction.

Administrator: superuser tasked with the responsibility of managing events and other users of the system.

CRUD: refers to the extend operations that the user (Administrator in this case) may perform.

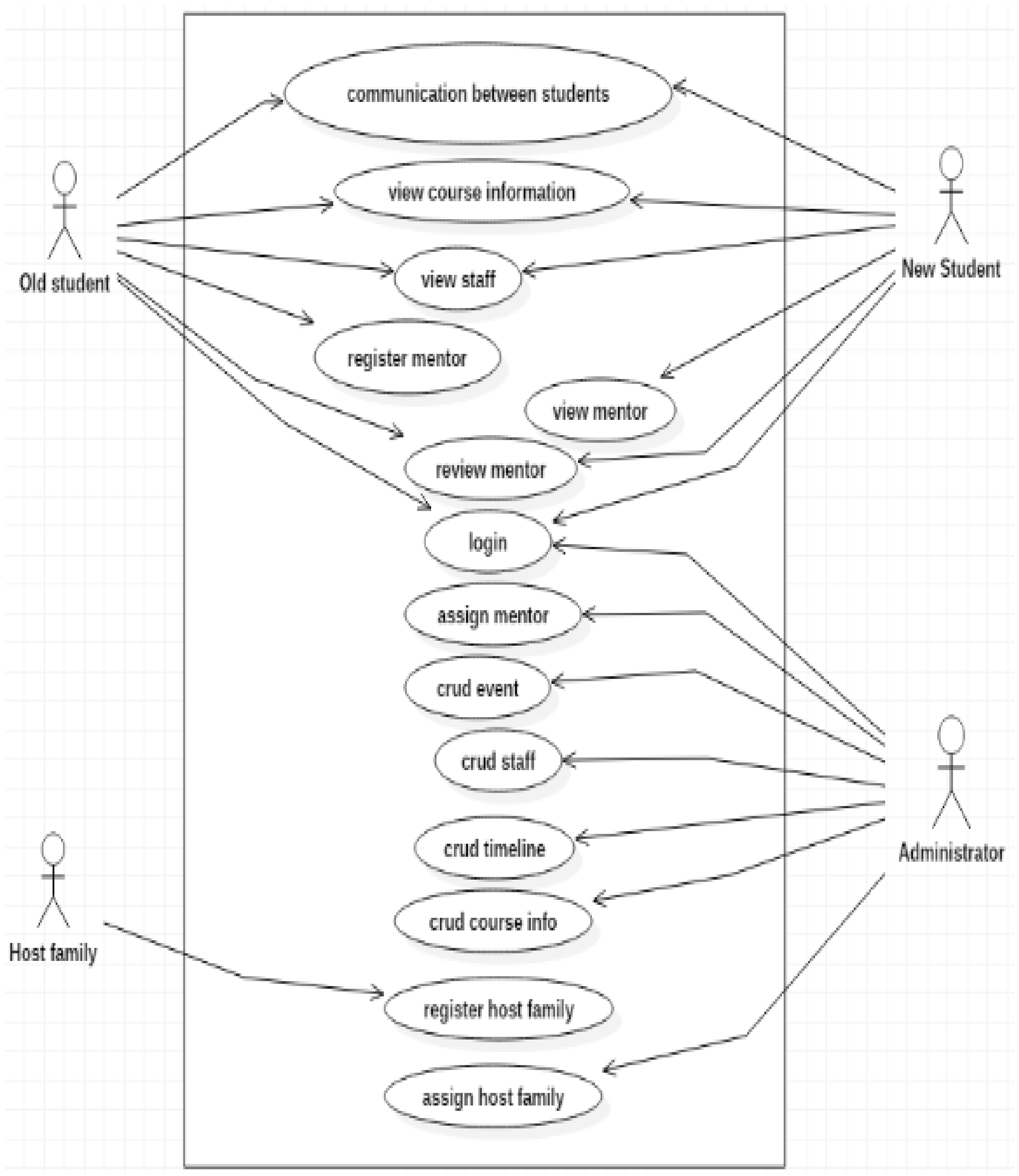


Fig 1.1 Use Case diagram

2.3 Use case descriptions

Use case ID	UC-1
Use case name	Register Mentor
Participating actor(s)	Mentor
Flow of events - 1. 2. 3. 4. 5. 6. 7. 8. 9.	Main flow 1. Mentor (old student) logs in to G-HELP system. 2. G-HELP system checks for the predefined criteria for becoming mentor. 3. System displays option to register as mentor upon meeting set criteria. 4. Mentor (old student) clicks the "Become a Mentor" button. 5. System redirects to Mentor Registration Form page. 6. Mentor (old student) fills form. 7. Mentor clicks on Submit button. 8. System submits completed form and registers old student as mentor. Use case succeeds
Exception Flow: 10.	2.a If predefined criteria is not met, system does not provide student with the option to become a mentor.
Entry condition:	Mentor (old student) must be logged in to the system and meets the criteria of being mentor.
Exit condition:	System informs the user that registration is successful.
Quality:	Response time for registration form submission should be within 30 seconds.

Fig 1.2 Detailed description of the Register Mentor use case

Use case ID:	UC-2
Use case name:	Assign Mentor
Participating actor(s):	Admin
Flow of events:	Main flow 1. Admin clicks on assign mentor menu on the admin page 2. G-HELP system redirects admin to assign mentor page 3. Admin views preferences of new students and preferences of registered mentors to see a match 4. Admin selects a student and a mentor based on the preferences 5. Admin clicks the assign mentor button

	6. System provides message that the assignment is successful Use case succeeds
Entry condition:	Admin must be logged in to the system and be on the assign mentor page
Exit condition:	System informs Admin that "Assign Mentor" operation is successful.
Quality:	System should not allow the assignment of more than one mentor to new students at the same time. Admin session should timeout after an inactivity period of 300 seconds

Fig 1.3 Detailed description of the Assign Mentor use case

Use case ID:	UC-3
Use case name:	Add Event
Participating actor(s):	Admin
Flow of events:	Main flow 1. Admin clicks the "Add Event" menu. 2. System provides Admin with add event form. 3. Admin fills the details of the event. 4. Admin clicks on Submit button. 5. System provides message that event addition is successful. Use case succeeds
Exception flow	5.a System fails to add event if event date chosen is already in the past
Entry condition:	Admin must be logged in to the system and be on the add event page
Exit condition:	System informs Admin that "Add Event" operation is successful.
Quality:	System should not allow posting of event where date given is already past.

Fig 1.4 Detailed description of the Add Events use case

2.4 Class diagram

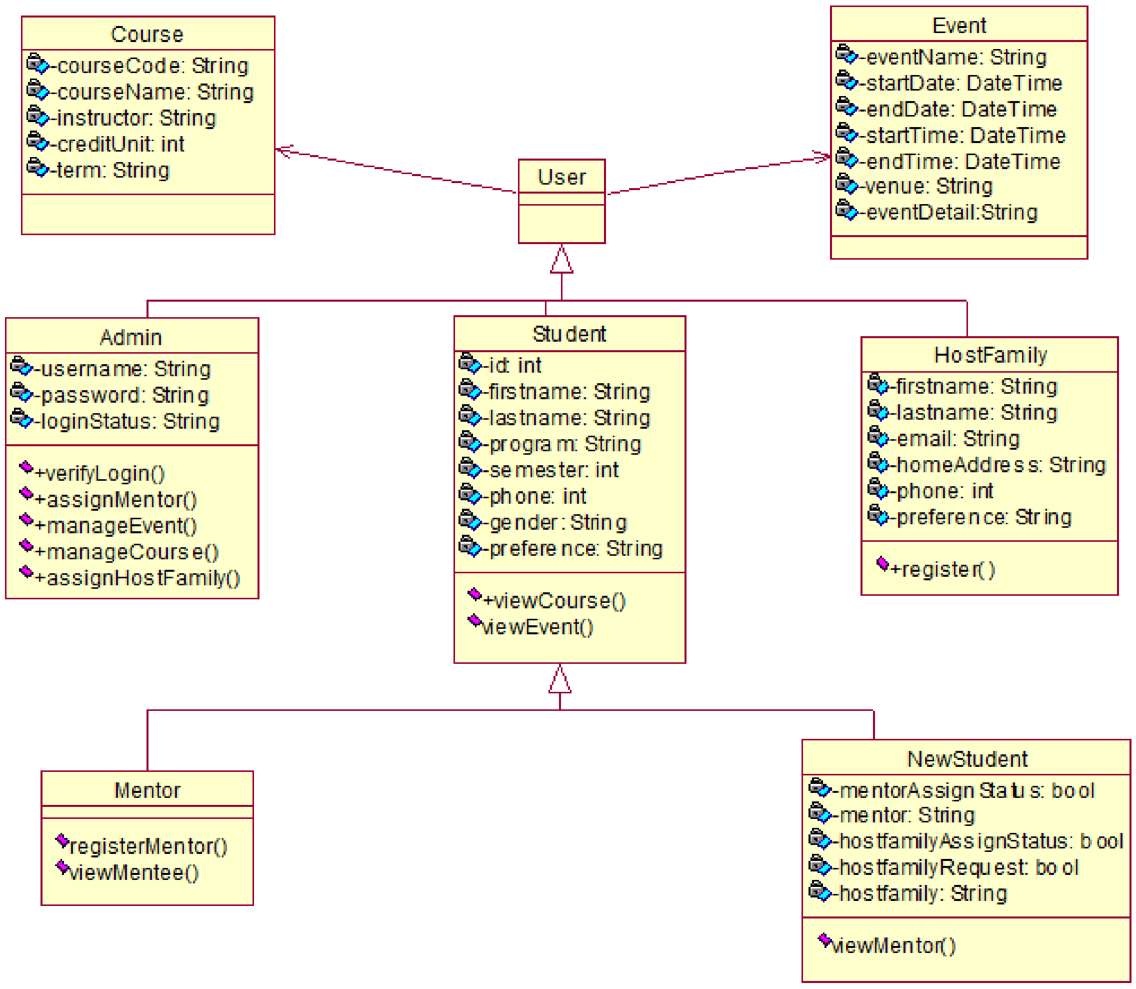


Fig 1.5 Conceptual Class Diagram (Domain Modeling)

2.5 Sequence diagram (dynamic model

Entity, boundary and Control

1. **Register Mentor -**

   Entity object – Old Student

   Boundary object - Login Button, BecomeMentorButton, RegisterMentorForm, RegisterMentorAcknowledgement

Control object - LoginButtonControl, RegisterMentorControl

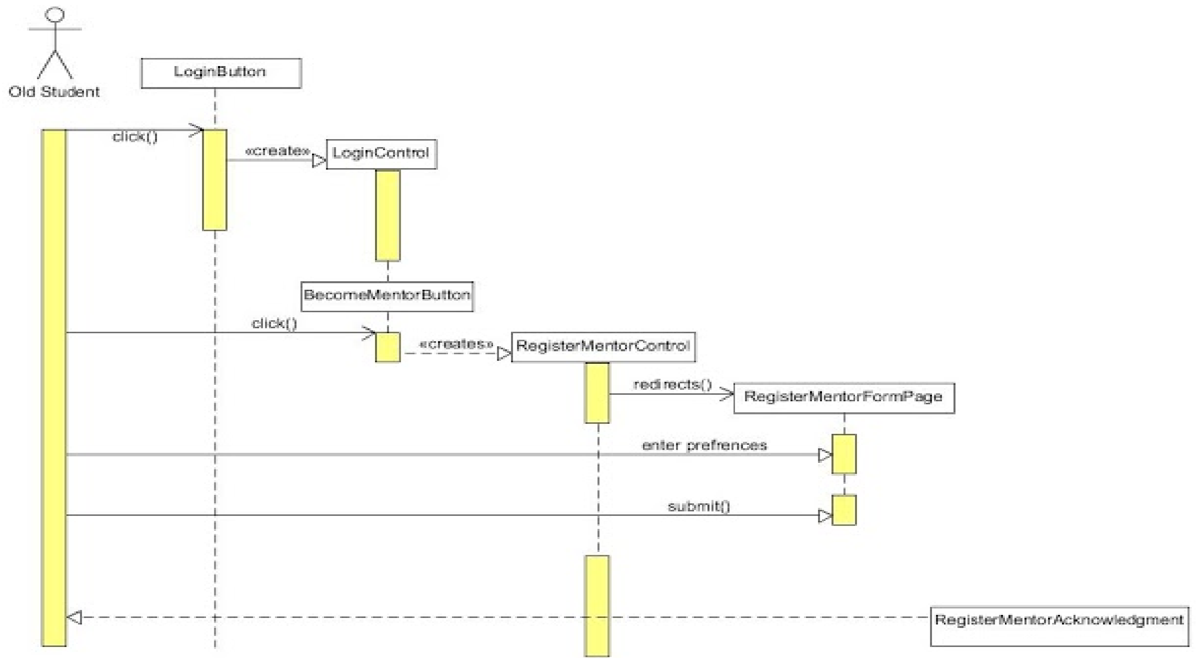


Fig 1.6 Sequence Diagram - Register Mentor

2. Assign Mentor -

Entity object - Administrator

Boundary object - AssignMentorForm, AssignMentorAcknowledgment, AssignMentorMenu

Control object - AssignMentorControl

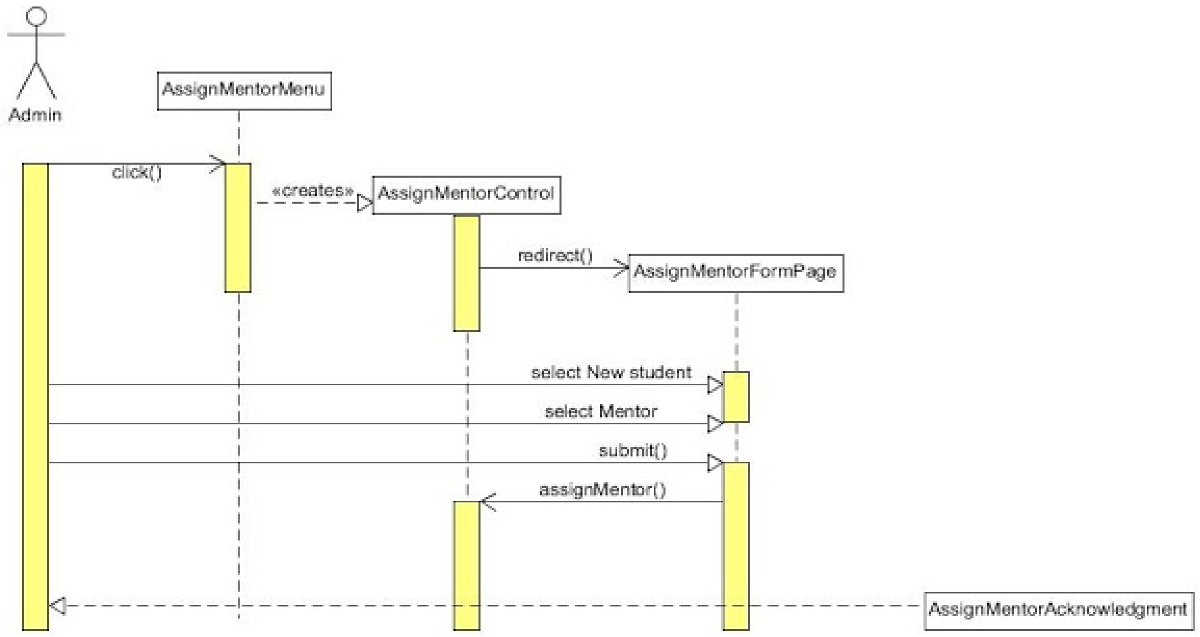


Fig 1.7 Sequence Diagram - Assign Mentor

3. AddEvent -

Entity object - Administrator

Boundary object – AddEventForm, AddEventMenu, AddEventAcknowledgment

Control object - AddEventControl

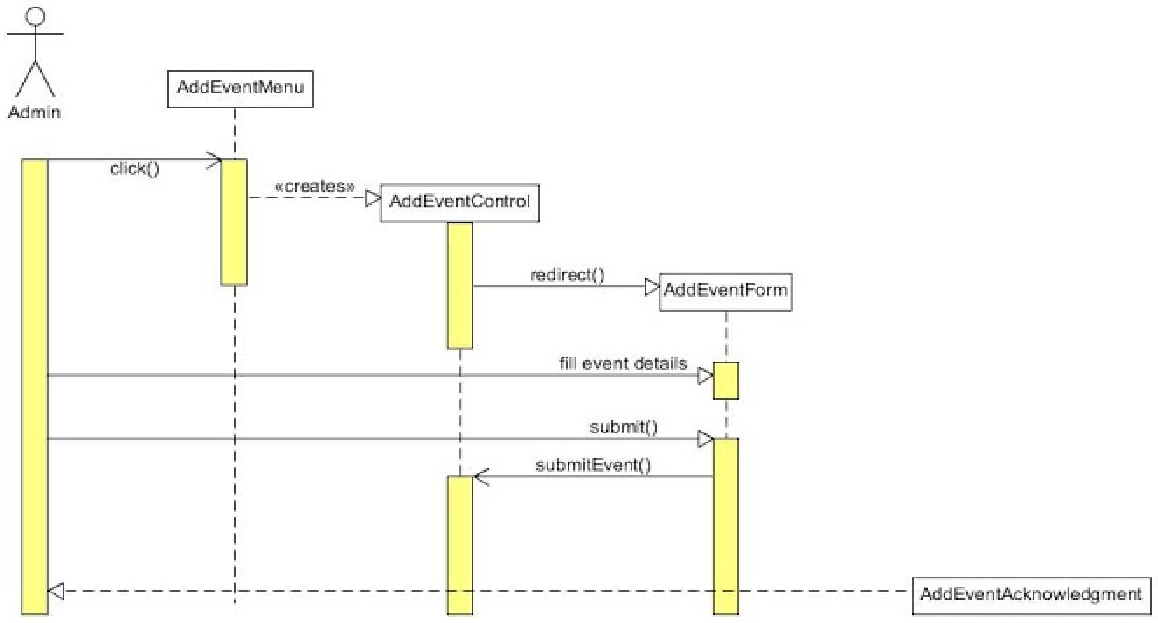


Fig 1.8 Sequence Diagram - Add Event

2.6 Design goals

2.6.1 Low operating cost

The cost of running the system should be minimized. For this reason, we will make use of open source components. Examples include node.js, express.js and MongoDB

2.6.2 Performance

We want to be able to process as many requests from the web browser as possible. To achieve this, we have chosen to use the Node.js together with Express.js to build a web server that optimizes response time and throughput.

2.6.3 Scalability

We anticipate much stress on the G-HELP system as the number of users grows as well as increase in the storage data. So, we have chosen a scalable database management system–MongoDB.

2.6.4 Security

We intend to keep the system as secure as possible. So, we shall be implementing controls to ensure that users of the G-HELP system (students, mentors, host family) only have access to unauthorized subsystems.

2.6.5 Ease of use

The functionalities of the G-HELP system will be such that each user can easily make use of them. For example, the frontend functionalities like 'register mentor' and administrative tasks such as assign mentor, add events will require simple button clicks and simple form-filling, as applicable.

Developers	**Clients**	**End Users**
1. Performance 2. Ease of use 3. Security	1. Scalability 2. Low Operating cost	1. Ease of use

2.7 Addressing the design goals

2.7.1 Mapping subsystems to components

G-HELP is a web application that supports multi-users. Users will access the system through a web browser which is installed on their machines (e.g. PC). The G-HELP subsystems will need a Web Server to process different users' request. We select the Node.js, a JavaScript runtime built on chrome's V8 JavaScript engine as our implementation technology alongside express.js framework. The Node.js provides us with an event-driven, non-blocking I/O architecture in addition to a Web Server built on express.js framework which addresses our performance goal. The Node.js alongside the express.js framework, being an open-source, also helps us to address another of our design goal: low operating cost.

2.7.2 Identifying Persistent Objects

Different sets of objects will be created within the subsystems of the G-HELP system. For example, we have to keep information about different mentors registered on G-HELP. These objects created need to be persistent and for that reason we need a database management system to help us manage these persistent objects. We select the MongoDB, a NoSQL database which is not only ideal for our implementation technology (Node.js) but offers us easy scalability as our data objects grow.

2.7.3 Providing access control

As much as we can, we want to ensure that different actors of G-HELP only have access to pages for which they are authorized. For example, the admin subsystem will be secured from other actors and subsystems of G-HELP. The admin only will have the privilege to assign mentor and add events. Mentors should be able to see "reviews" by students but should not be able to review themselves, which is a function meant for the new students (mentees).

2.7.4 Designing the global control flow

Using Node.js provides us with a different control flow modules. The callbacks function and event module of Node.js helps to process multiple requests. As a result, our Web server which will be built using Node.js with express will receive requests from the web browser and processes using different control-flow module depending on the kinds of requests. The response time and throughput will be optimized using the control-flow module of Node.js.

2.8 Object modeling

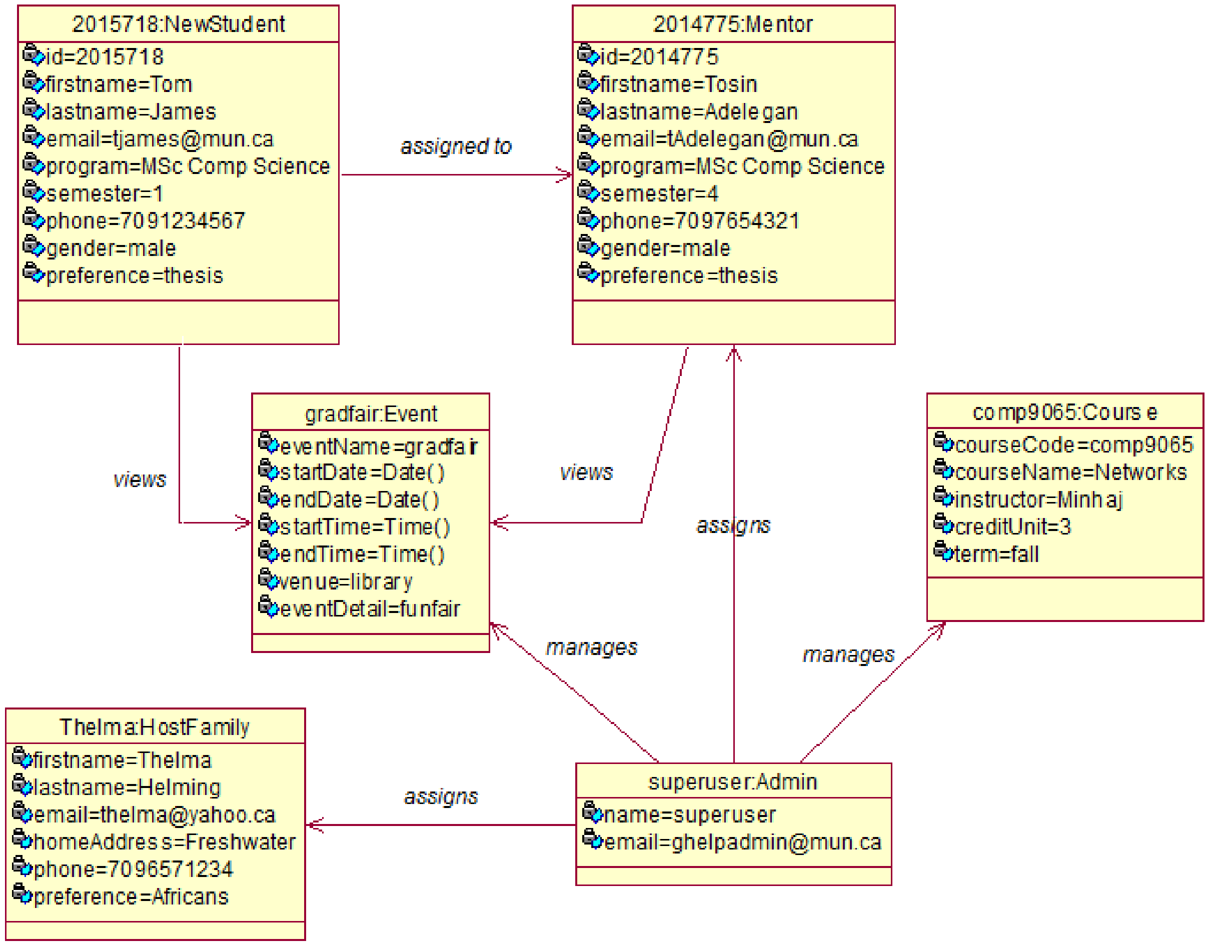
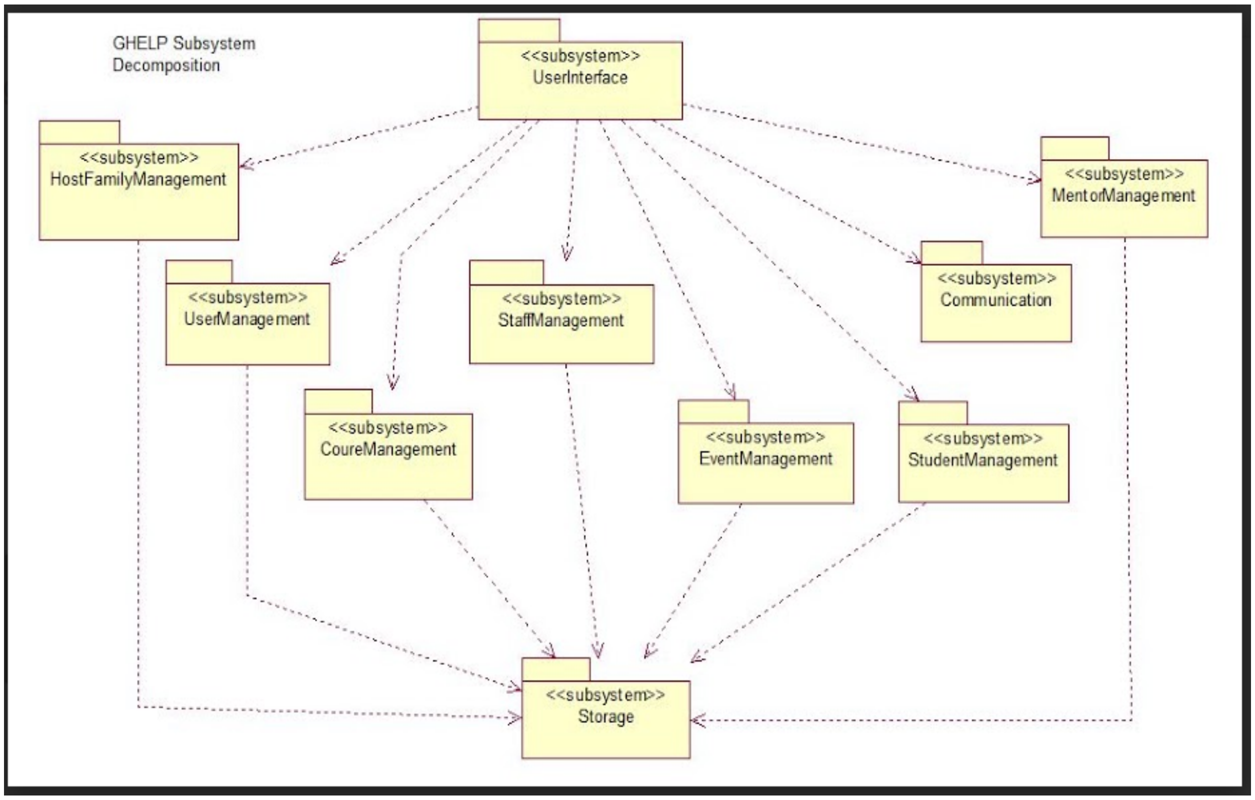


Fig 1.8 object model

2.9 System decomposition



GHELP Subsystem Decomposition
<<subsystem>> UserInterface
<<subsystem>> HostFamilyManagement
<<subsystem>> MentorManagement
<<subsystem>> UserManagement
<<subsystem>> StaffManagement
<<subsystem>> Communication
<<subsystem>> CoureManagement
<<subsystem>> EventManagement
<<subsystem>> StudentManagement
<<subsystem>> Storage

Fig 1.10 G-HELP subsystem decomposition

2.10 System architecture

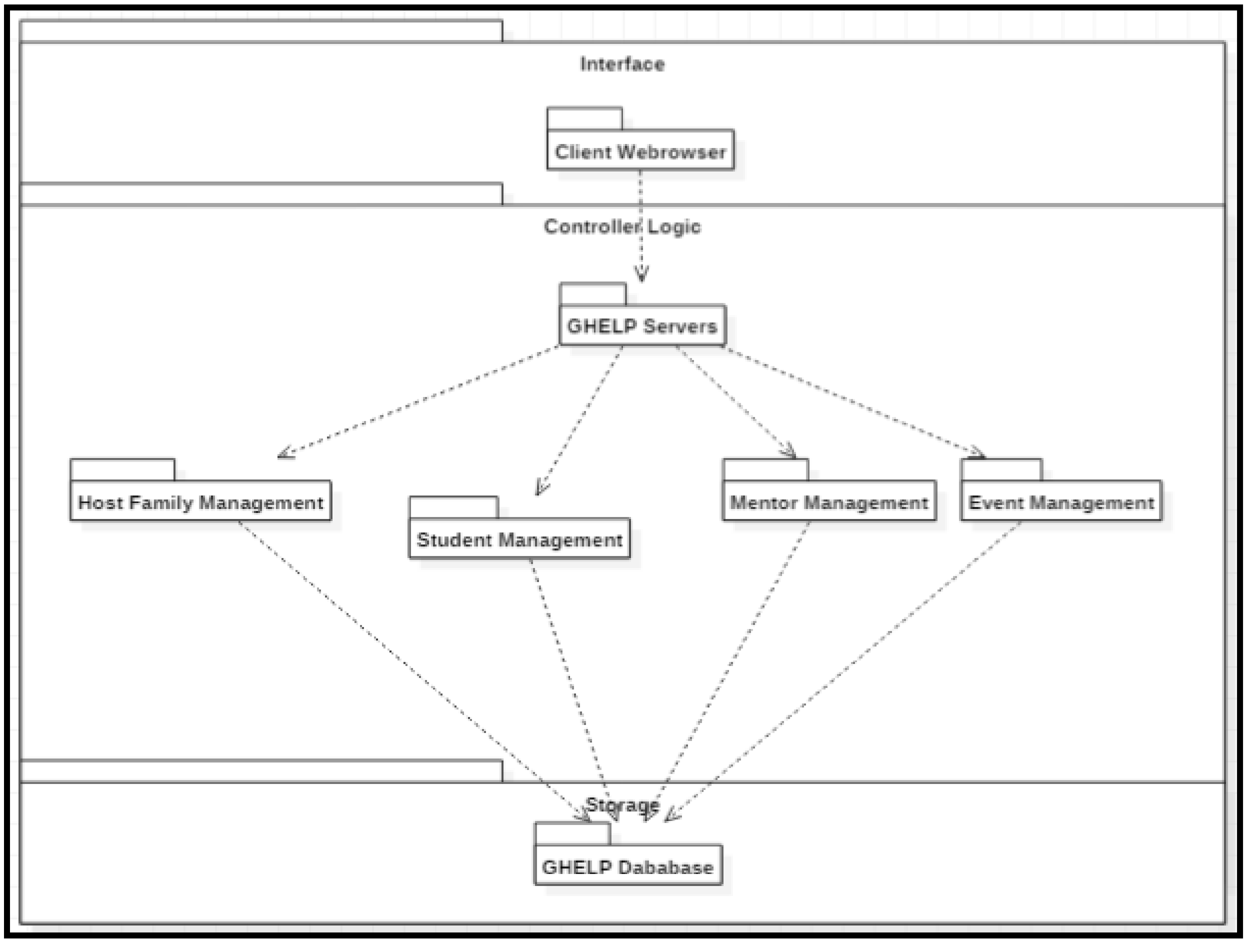
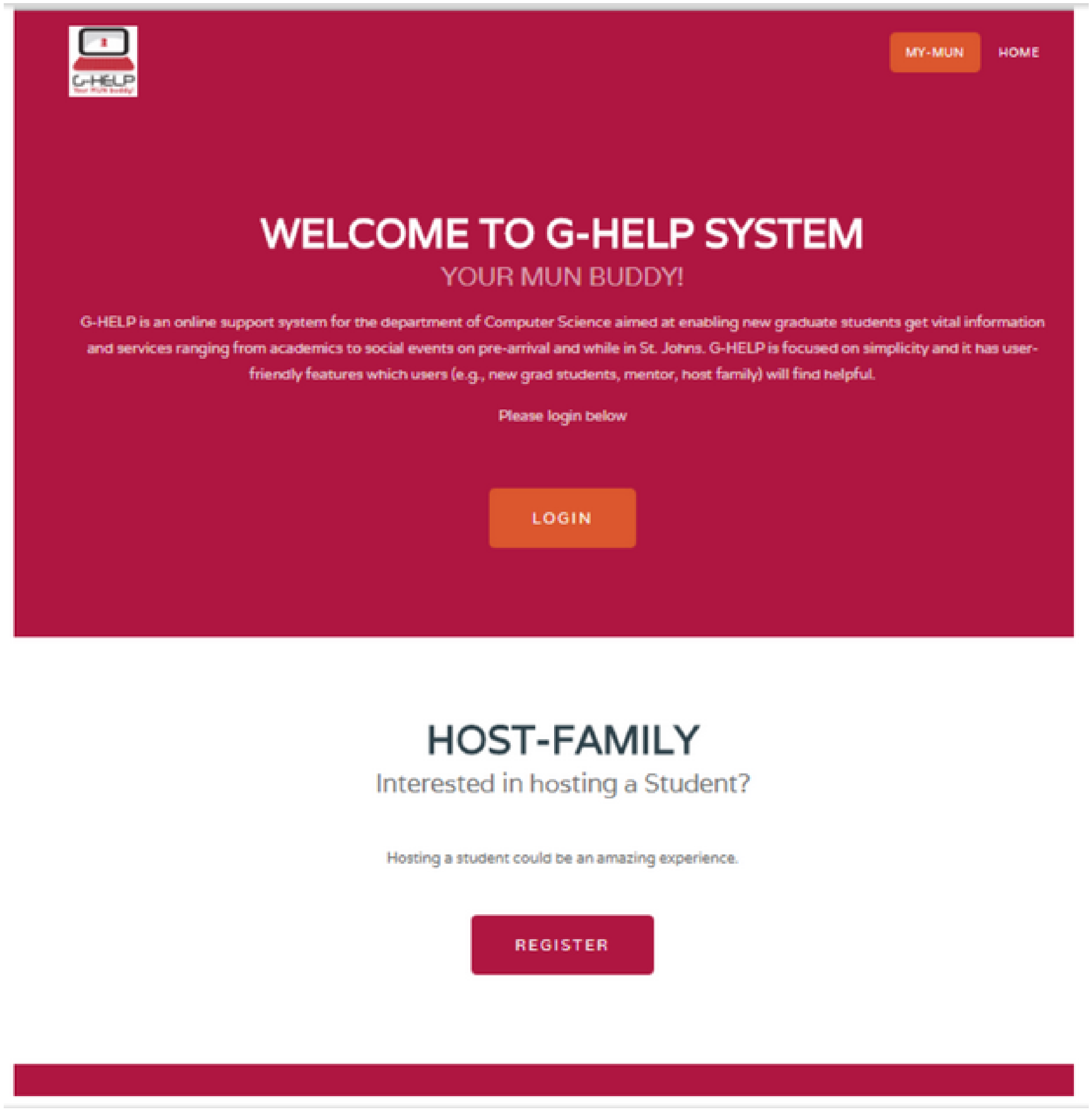


Fig 1.11: Three-tier Architectural Style

3.0 System Prototype

3.1 G-HELP system interfaces



WELCOME TO G-HELP SYSTEM

YOUR MUN BUDDY!

G-HELP is an online support system for the department of Computer Science aimed at enabling new graduate students get vital information and services ranging from academics to social events on pre-arrival and while in St. Johns. G-HELP is focused on simplicity and it has user-friendly features which users (e.g., new grad students, mentor, host family) will find helpful.

Please login below

HOST-FAMILY

Interested in hosting a Student?

Hosting a student could be an amazing experience.

Fig 1.12: Home page

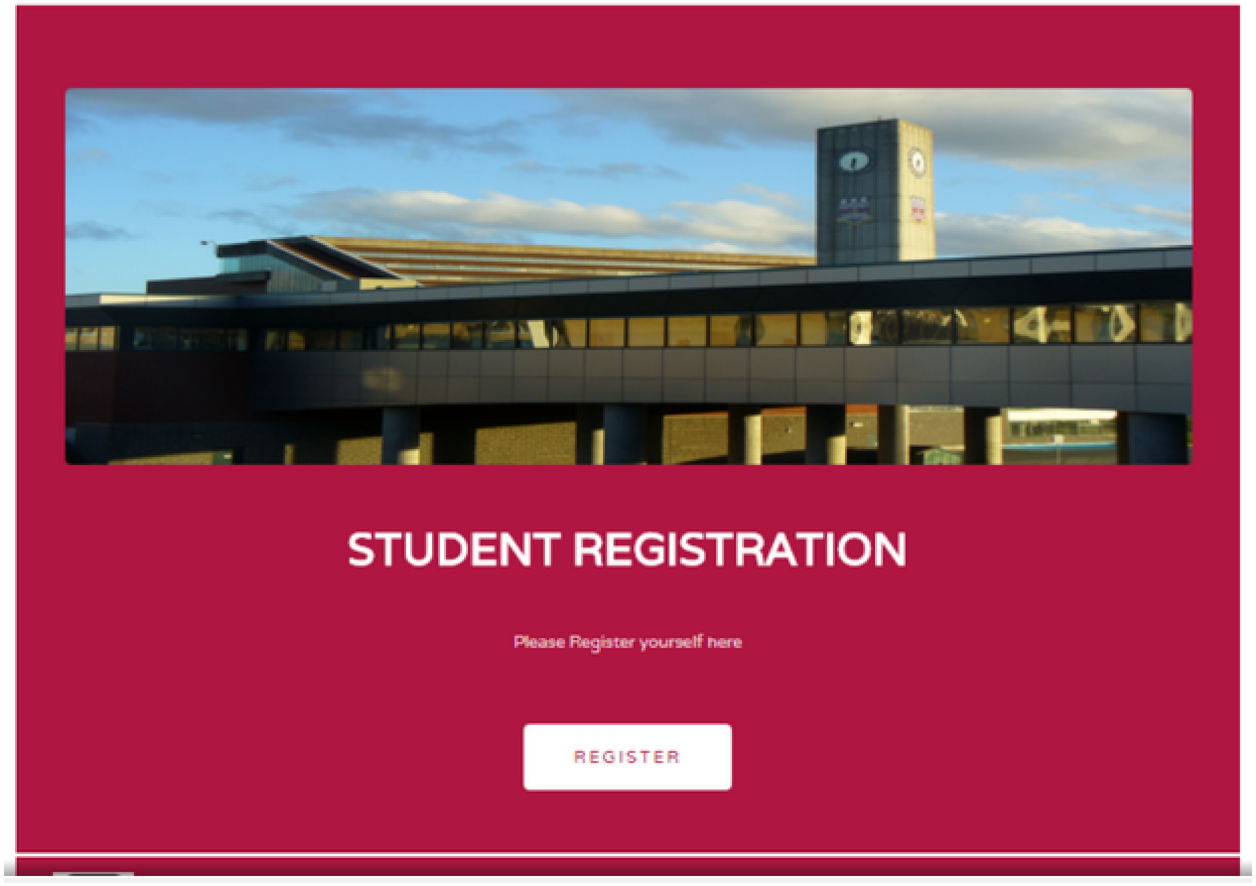
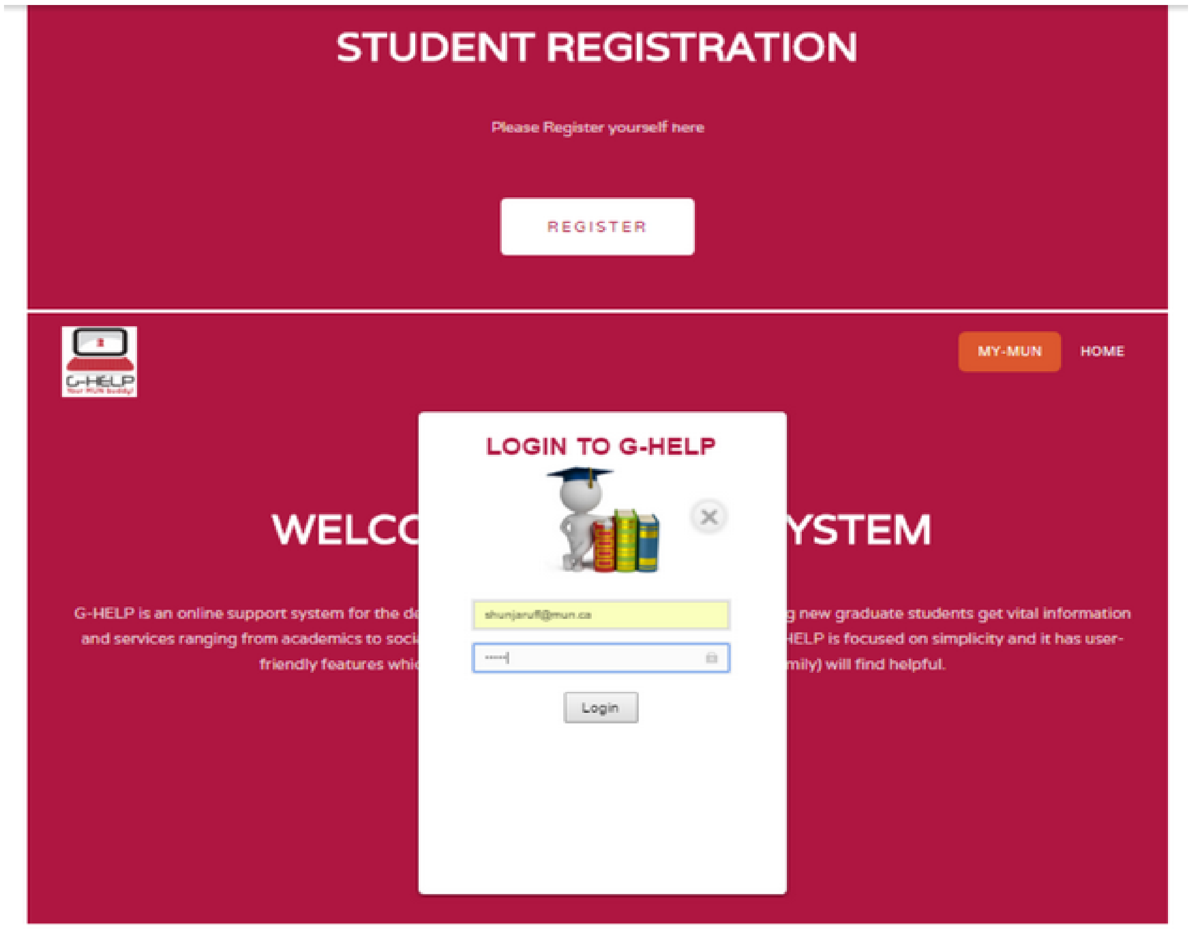


Fig 1.13: Student registration page


STUDENT REGISTRATION

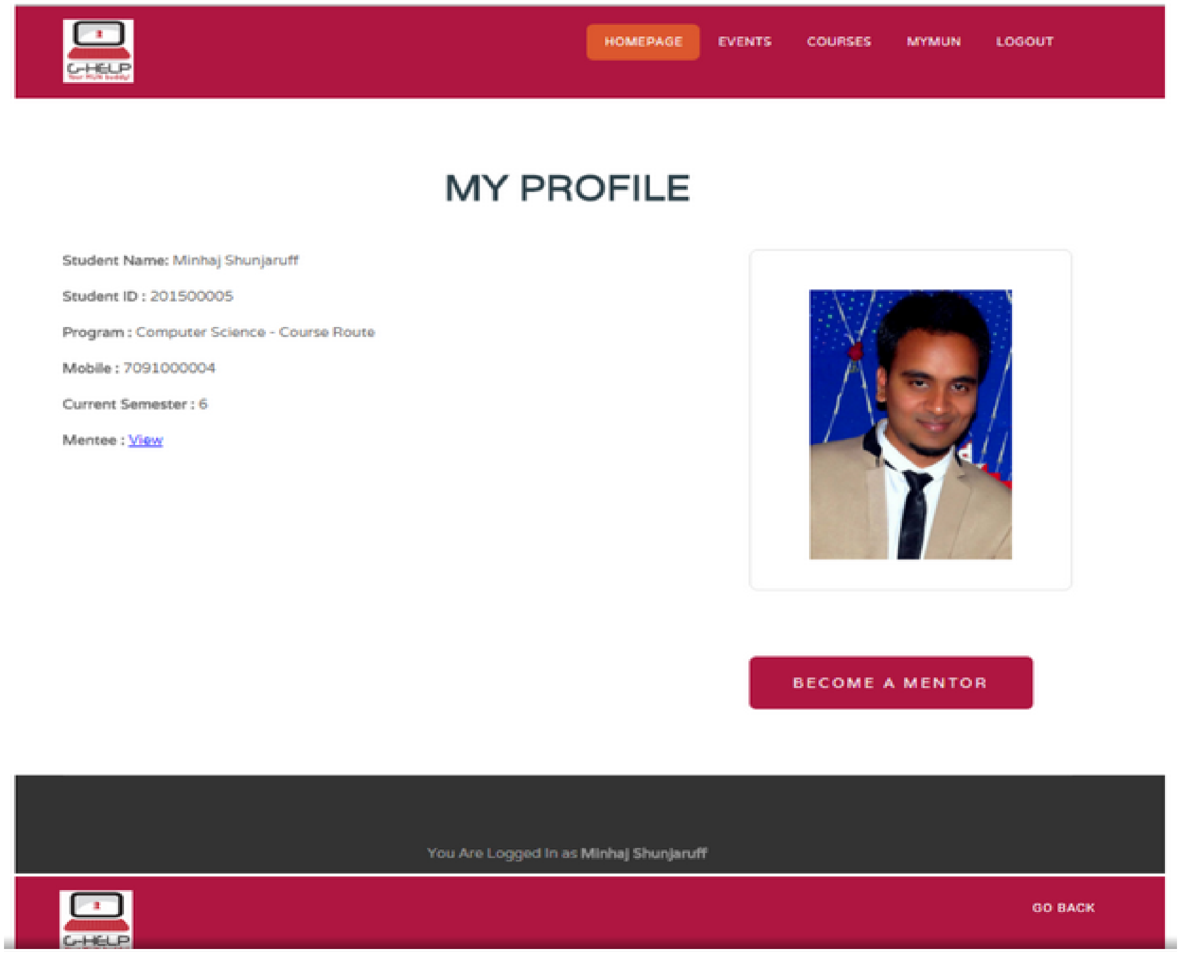
Please Register yourself here

WELCO YSTEM

G-HELP is an online support system for the de g new graduate students get vital information and services ranging from academics to soci HELP is focused on simplicity and it has user-friendly features whi mily) will find helpful.

shunjaruf@mun.ca

Fig 1.14: login page



MY PROFILE

Student Name: Minhaj Shunjaruff

Student ID : 201500005

Program : Computer Science - Course Route

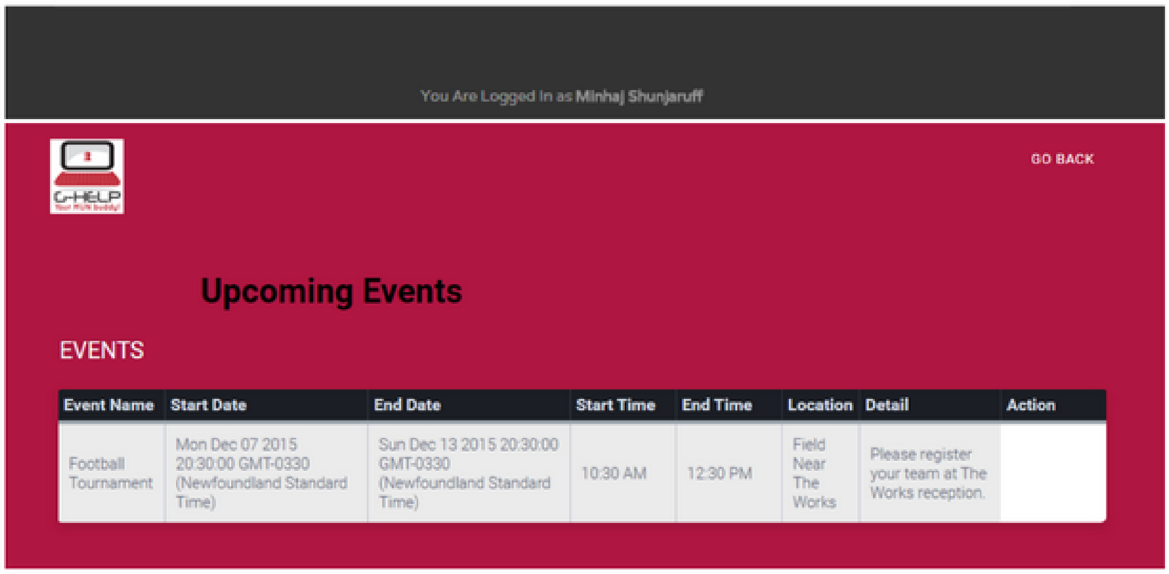
Mobile : 7091000004

Current Semester : 6

Mentee : View

You Are Logged In as **Minhaj Shunjaruff**

Fig 1.15: Mentor page


You Are Logged In as Minhaj Shunjaruff

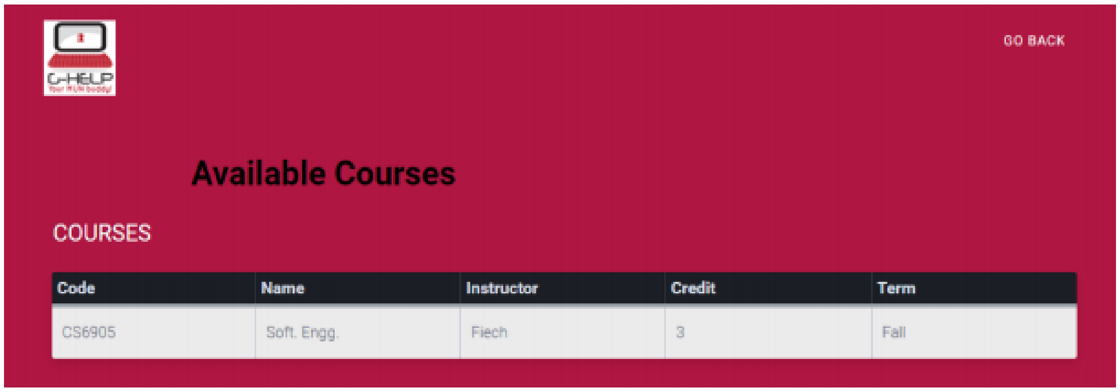
GO BACK

Upcoming Events

EVENTS

Event Name	Start Date	End Date	Start Time	End Time	Location	Detail	Action
Football Tournament	Mon Dec 07 2015 20:30:00 GMT-0330 (Newfoundland Standard Time)	Sun Dec 13 2015 20:30:00 GMT-0330 (Newfoundland Standard Time)	10:30 AM	12:30 PM	Field Near The Works	Please register your team at The Works reception.	

Fig 1.16: Events page

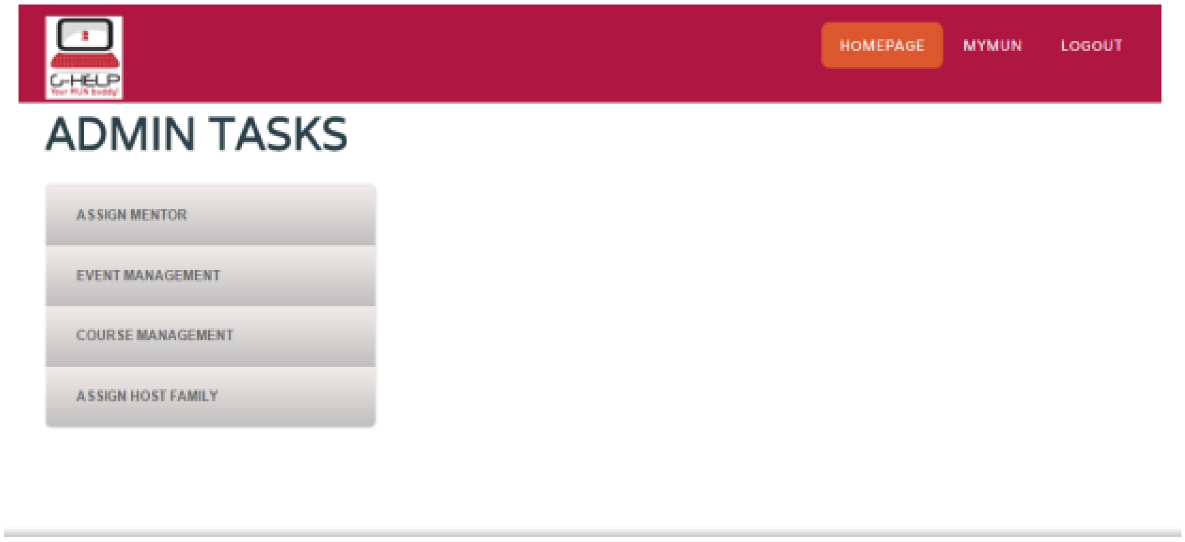

GO BACK

Available Courses

COURSES

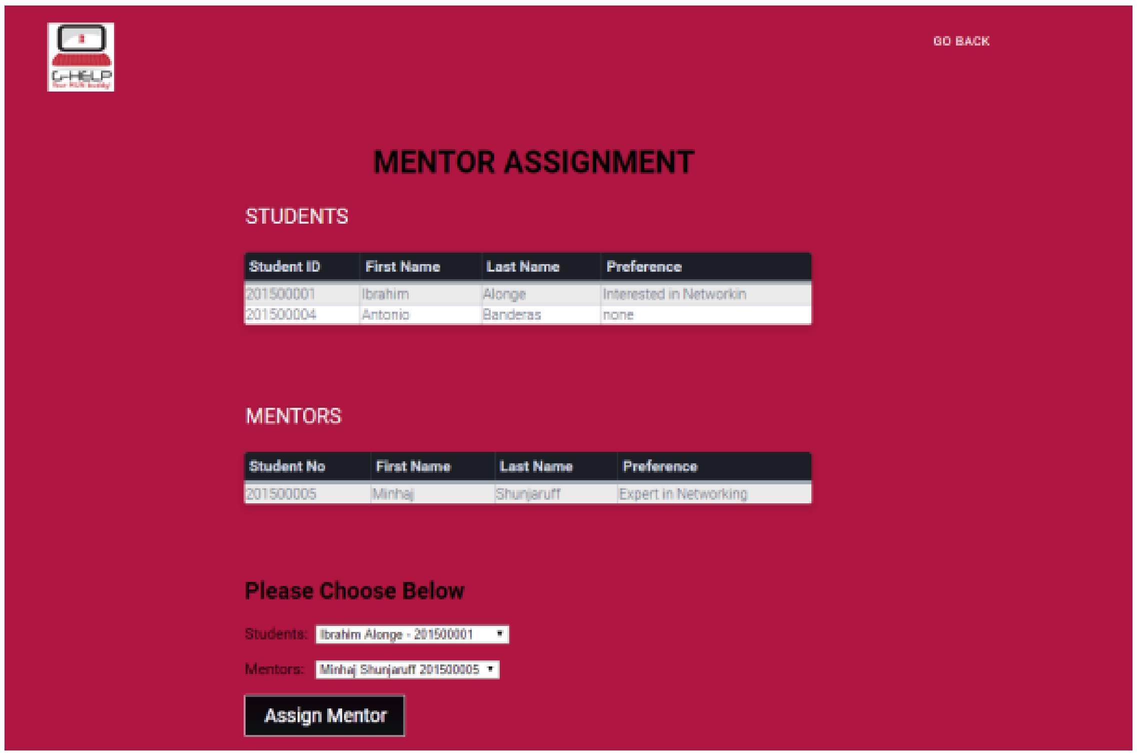
Code	Name	Instructor	Credit	Term
CS6905	Soft. Engg.	Fiech	3	Fall

Fig 1.17: course view page



ADMIN TASKS

Fig 1.18: admin page



MENTOR ASSIGNMENT

STUDENTS

Student ID	First Name	Last Name	Preference
201500001	Ibrahim	Alonge	Interested in Networkin
201500004	Antonio	Banderas	none

MENTORS

Student No	First Name	Last Name	Preference
201500005	Minhaj	Shunjaruff	Expert in Networking

Please Choose Below

Students: Ibrahim Alonge - 201500001

Mentors: Minhaj Shunjaruff 201500005

Fig 1.19: Mentor assignment page

4.0 Project review

4.1 Challenges faced

The greatest challenge we encountered in developing this system was to learn a new programming language in a short space of time. It was a daunting experience; one that practically overwhelmed us, given that we had other academic demands in the other courses that we were taking in the semester. At the end of the day, the gains realized in terms of experience and skills acquired were worth the efforts put into undertaking the project. Just as the popular saying goes, that necessity is the mother of inventions, the pressure to deliver on the project motivated all the resources we could muster to develop the G-HELP system within schedule.

4.2 Lessons learnt

First, we were able to learn a new language in node.js, at least to some extent. In addition, we were able to learn to use Ajax JQuery form post, as a necessity to implement some desired features on the system. Working in a group with people having diverse skills and from different cultures, we have learnt some significant aspects of project management, including planning, task division, working under short deadlines and changing user requirements.

4.3 Conclusion

The G-HELP system we have developed has been challenging and interesting to develop at the same time. From the outset when we did requirement elicitation, user requirements have been in constant natural change, but the application of the industry's best practices such as Rationale Management and Configuration Management have helped us to stay on course and deliver a useful and usable software with quality features such as simplicity and scalability. It is hoped that the system will find usefulness in the hands of enthusiasts in the node.js programming language domain, and that enhancements can be made in the future to improve the functionalities of the system.