'perpetually available' academic advisor specifically designed to handle cases of students who are unable to physically meet with a human advisor [11]. A similar system was put forward by reference [10] using Java Expert System Shell (JESS) and XML.

The drawback of both systems however is that students are required to supply solutions with a significant amount of information for results to be generated. This can sometimes result in the possibility of students entering inaccurate information and hence getting inaccurate advice, or students becoming disinclined to use such a system as they would prefer a system less demanding of them.

3) Advanced Automated Systems (AAS): In an attempt to make solutions more 'student friendly', we sought to turn attention to systems that produced similar results to IIAS with less student interaction. A subset of these was termed Advanced Automated Systems, which utilized prescribed algorithms and computational power to generate advice based on existing data.

Reference [4] discuss a system which uses database queries on the information stored within each student's transcript in the SIS in order to give students advice as to what courses they should take in the next semester as well as give their graduation status. Another system surrounding PHP, MySQL and Email technology is presented by reference [7] which again uses database queries to group all related student and course information for the purposes of generating a list of suggested courses to be taken in the next semester. A similar system is also shown in reference [13] whereby developers used Wxpython alongside an access database to deploy a desktop solution to facilitate postgraduate students.

4) Intelligent Advanced Automated Systems (IAAS): AAS can be quite effective and produce satisfactory results. However, the database queries used to generate such results can be quite complex and resource intensive. Furthermore, the rules within advising solutions that govern programme structure and university regulations are frequently reviewed, which can then require the need to modify the SQL queries which act as rules governing the system's functionality. A more practical approach, promoting change and easy maintenance would be using expert system technology to manage the rules that govern the advising system. Such systems that provide this functionality, similar to that of AAS are coined as Intelligent Advanced Automated Systems and research showed that this was the most favoured approach when creating advising solutions for institutions.

Such systems were observed to use various reasoning strategies to achieve their appropriate results. One for example showed how Case-Based Reasoning (CBR) was used to develop a system that recommended a suitable major to students based on comparing their student information against similar historical cases [9]. The system was proven to be quite effective when advising students who were reading for general degrees, provided there were sufficient historical cases within its knowledge base (KB). Rule-Based Reasoning (RBR) systems were also designed whereby developers used Forward and Backward Chaining procedures in order to generate appropriate advice. Such capabilities as well as a 'cognitive and emotional filter' were used in a solution proposed by reference [1] in order to provide course advising to students. "IS-Advisor" also followed a rule-based approach alongside its Object Oriented Database [6]. Other systems such as the "Course Advisory Expert System" went a step further by providing both reasoning capabilities in order to facilitate an even higher quality of advising when prescribing course recommendations [8].

With the emergence of ontologies and semantic web technology, we saw that developers harnessed ontology driven methodologies to tackle the dynamic and complex nature of student academic planning and scheduling by creating "E-Advisor", a multi-agent intelligent advising system [3]. Designed for the Master of Science in Information Systems in Athabasca University in Canada, it allowed students the ability to add preferences of specialization to their profile and then recommend courses based on these preferences. Its multi-agent nature also made system maintenance easy, as it allowed the use of other agents while updating others making it a highly available system, and possibly one of the better intelligent web-based advising systems in the world today.

B. AdviseMe as opposed to previously implemented solutions

After much scrutiny of the problem domain as well as the solutions discussed in section II(A), it was clear that a system with an IAAS architecture be the optimal solution for enhancing the level of advising currently experienced in FST, as it would seek not only to minimize the need for student input and generate satisfactory results, but also significantly reduce human advisor workload, among other benefits. This places "E-Advisor" as the top candidate for consideration. However, several issues arose which lent to the ultimate dismissal of such a solution.

While E-Advisor has proved its successful application at Athabasca University, it is noteworthy that the system was tailored to advise students in a single MSc Programme [3] and not a range of programmes that would be required of a system serving FST. Also, the system focuses on course scheduling, but no mention of the system handling other important features required by FST was made in the literature. Determining oral assessment qualifications and graduation status are but only some of the important student issues that human advisors often have to repeatedly solve.

As a result, AdviseMe's design was proposed since not only does it facilitate aforementioned services as well as tracking of student history and other useful features, but also facilitates both undergraduate and postgraduate programmes that can be managed on a department, faculty or even campus based level. Furthermore, AdviseMe's solution requires less staff involvement as it only needs a minimum of 1 administrator to configure the environment while E-Advisor's functionality is dependent on every instructor in the department for the proper working of the system. While this extended staff involvement poses benefits in the context of Athabasca University, it fails to alleviate the staffing issues faced in FST. For these reasons, as well as the fact that no single previously mentioned solution efficiently solves the issues outlined in section I(C), the design and implementation of AdviseMe was born.

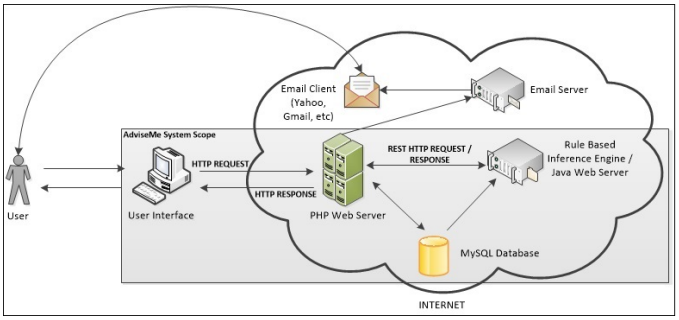


Fig. 2: Flow of data between modules within AdviseMe

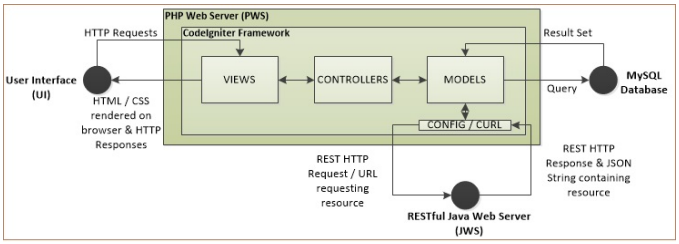


Fig. 3: Overview of the PHP Web Server (PWS) Architectural Design

III. System Design

A. Overview

AdviseMe typically has 3 main user groups (students, advisors and administrators) all of which interact with the system via a single browser rendered user interface.

When requests are made, they are passed to the PHP Web Server (PWS) which acts as the mediator of all data flow within the system. The system provides both 'intelligent' services (eg. Course Advising) as well as 'non-intelligent' services such as system management and email communication. For all intelligent services, the PWS communicates with the RESTful Java Web Server (JWS), which handles all intelligent processing via its use of ontologies and its rule based inference engine.

The PWS also sends requests to the Email Server for disseminating messages to users based on the services accessed. For example, when an administrator updates a rule in the system, the PWS would send a request to the Email Server in order to broadcast a message to advisors, notifying them of changes made.

All other facilities provided by the PWS are fuelled by information retrieved from or sent to the MySQL database via the UI. The JWS also retrieves information from the MySQL database before processing data and sending results to the PWS. Ultimately, the PWS would manage all processing within the system and render the output in a sleek, intuitive form via the UI.

B. PHP Web Server (PWS)

The PWS was designed and implemented using the CodeIgniter Framework and facilitates the server side implementation of the web application, as well as the interface for communication with the Email Server (Google's Gmail SMTP Server), MySQL database and RESTful JWS (by means of the Curl URL Library). CodeIgniter's extensive documentation and 'Model-View-Controller' architectural style promoted quick and easy implementation of the PHP based application server. Controllers accept data from the models and pass them to views which render the information in an intuitive format for end users. In AdviseMe, the models handle two types of data requests, the first being calls to the MySQL database using CodeIgniter's Database Library and the other being requests being sent to the RESTful JWS. Information from the JWS is retrieved via Uniform Resource Locators (URLs) and the use of the Curl URL Library, for the consumption of REST services. An overview of the PWS architecture is shown in Figure 3.

C. RESTful Java Web Server (JWS) Design

Although not directly accessed by users, the JWS is essentially the driving force behind AdviseMe's core functionality. Using Apache Jena alongside ConnectorJ to create an ontology-based reasoning environment, it proves to be an ideal reasoning engine when seeking to determine non-trivial inferences on a student's academic record. These inferences are then exposed as JSON objects via RESTful web services, made possible by using tools such as Jersey and JSON-Simple.

Jena's API provides a wide collection of classes and interfaces for the management of 'OWL-Based' technologies. OWL, an acronym for 'Web Ontology Language', is typically an extension of the Resource Definition Framework Schema (RDFS) which in turn is an extension of the Resource Definition Framework (RDF). RDF forms the foundation of how resource information should be structured, with RDFS and OWL enhancing the ways in which resources are described.

With RDF being a suitable standard for structuring data and OWL having an extensive vocabulary that can be easily interpreted by machines, Jena's Ontology API was used to create a simple ontology termed 'AdviseMeOnt' to be used within our system context. Comprising of a set of resources / classes immediately surrounding the academic advising environment and their appropriate properties, it was used to model all information within the student's transcript in order to produce meaningful inferences to support the advising process. A listing of some of the concepts within 'AdviseMeOnt' is given in Table 1.

Student Profiles were then created by extracting information from the MySQL database to create and populate an ontology model using the Jena Ontology API, and then exporting the information to a RDF file.

In addition to creating RDF-Based student information, the JWS was also designed to extract information from the database and create a set of user defined rules to be applied to the RDF files. These custom rules included prerequisite and exemption rules, as well as rules containing other user defined variables. The JWS also can generate a set of fixed, system-defined rules by calling the appropriate functions within the server.

Once RDF-Based student profiles and the set of executable rules are existent, the JWS can then generate non-trivial

TABLE I: Listing of some concepts and properties within 'AdviseMeOnt'

advMe:COURSE
advMe:COURSECODE
advMe:COURSENAME
advMe:SEMESTEROFFERED
advMe:MARKOBTAINED
advMe:NUMBERCREDITS
advMe:PROGRAMME
advMe:PROGRAMMENAME
advMe:LEVELONECREDITS
advMe:CORECREDITS
advMe:ELECTIVECREDITS
advMe:FOUNDATIONCREDITS
advMe:HASCORECOURSE
advMe:HASELECTIVECOURSE
advMe:HASFOUNDATIONCOURSE
etc...
advMe:STUDENT
advMe:STUDENTID
advMe:STUDENTNAME
advMe:STUDENTTYPE
advMe:ACADEMICSTANDING
advMe:FUNDINGSTATUS
advMe:CANGRADUATE
advMe:COMPLETEDCOURSE
advMe:ALLOWEDCOURSE
advMe:LOCKEDCOURSE
advMe:FAILEDCOURSE
advMe:POSSIBLEORALCOURSE
etc...

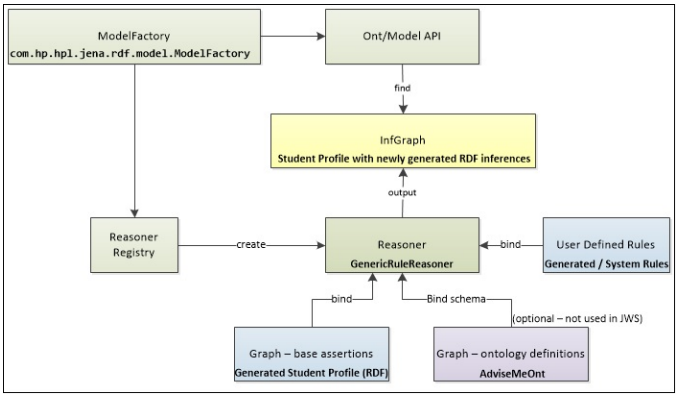


Fig. 4: Overall Structure of Inference Machinery used within Jena

inferences within the student profile using the Jena Inference Subsystem (JIS). This module, accessed via the Jena API, is designed to allow a wide range of inference engines to be integrated with Jena projects for the derivation of additional RDF data, by making inferences on existing RDF assertions. Figure 4 illustrates the overall structure of the inference machinery used within Jena and by extension the JWS.

The JWS accesses the JIS using the ModelFactory class to associate the RDF-Based student profiles with the appropriate reasoner. Jena provides a Generic Rule Reasoner which supports the user of user-defined rules as well as forward and

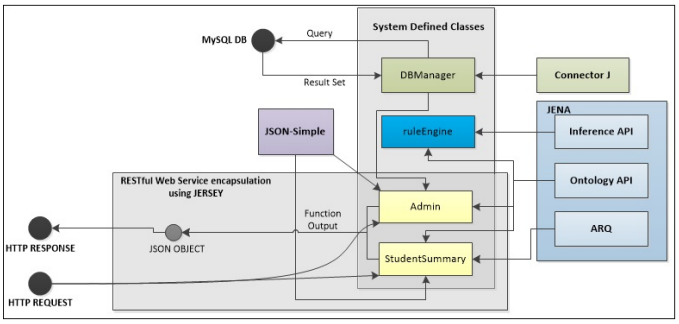


Fig. 5: Overview of the Java Web Server (JWS)Architectural Design

backward chaining capabilities, ideal for handling the ruleset within the JWS. The student profile and ruleset are bound to this generic reasoner and after processing, the reasoner outputs an InfGraph object. This is typically a new model of the student profile containing previously asserted as well as newly inferred information. The information is then saved to storage as an RDF file which is later parsed using the Jena Ontology API as well as ARQ (a query engine supported by Jena providing RDF querying capabilities) for extracting results.

In order to expose inference results to users in a presentable format such as via a structured website, information extracted from the InfGraph model was transformed into JSON format before being made available to external users via RESTful web services. This was done using JSON-Simple and Jersey Java packages respectively which allowed information to be passed to the PWS via an HTTP response.

As a result, the JWS communicates seamlessly with the PWS in order to make useful advising information for rendering via the UI. Administrators are also allowed to access the PWS via the UI in order to perform 'remote' tasks on the JWS such as re-creating a student's RDF-Based Profile and refreshing the user-defined rule base. Figure 5 shows an overview of architecture of the JWS.

IV. Methodology

A. Pre-Implementation Phase

Preceding system implementation, research was conducted by acquiring information from both students and advisors currently at UWI. Student's thoughts on a computerized advising system were collected via an anonymous questionnaire. Participants were selected from first year undergraduates straight up to postgraduate students in order to get a wide range of responses from those who would have been new, as well as quite accustomed to the traditional advisory process. Results from the questionnaire showed that the majority of students were dissatisfied with the traditional advising process, with 90% of them visiting advisors for the handling of general matters such as course scheduling, determination of graduation status and the handling of trivial student issues. Furthermore, students stated that they would expect such a system to offer 24/7 accessibility, course advising capabilities and support for handling other student issues such as holds and academic standing issues. They also suggested that the system possess

some level of intelligence so to give possibly better advice than existing advisors who are sometimes unaware of critical changes in the process.

Representing the advisory body was the Deputy Dean and Head of the Advisory unit at FST. In conclusion of her interview she acknowledged that an intelligent computerized system would have immensely benefited advisors by reducing their advisory load, promoting more dedication to special cases and enhancing the overall workflow of the advisory process. She verified the hypothesis that such a system would be able to handle a significant amount of student cases without the need for a human advisor and also highlighted that the ability of students to remotely contact human advisors if necessary could also help to streamline the other aspects of the advisory process. As a result of her confidence in the system, the design and implementation quickly came under way as it posed some value to UWI and more specifically FST.

B. Implementation Phase

As stated in section III, AdviseMe comprises of two distinct modules, the RESTful JWS which provides all intelligent processing with respect to generating academic advice and the PWS which converts this advice into a presentable format to be rendered via a web browser, among other functions. Section IV(B1) gives a summary of some of the services offered by the JWS while Section IV(B2) summarizes the functionality of the overall system as controlled by the PWS.

1) RESTful JWS Implementation: As stated before, the JWS provides RESTful services accessible via HTTP requests. These services comprise of the following:

- Creating / Refreshing the 'AdviseMeOnt' Ontology
- Creating / Refreshing the rule base (comprising of user and system defined rules)
- Creating / Refreshing all student profiles (based on transcript information in SQL database)
- Creating / Refreshing a single student profile(based on transcript information in SQL database)
- Generating advisory information for a given student (based on his student profile)

All of these services return a JSON string to the requesting agent after processing. With exception to the last service, all requests return a success value of "TRUE" if the request was successfully processed or "FALSE" otherwise (see Figure 6). In the case of generating the advisory information however, the request,once successful, returns a JSON object containing the following information:

- Student Name, ID and Student Type (Full Time / Part Time)
- Academic Standing and Funding Status (Whether or not the student can be covered for sponsorship payments)
- Number of years in current programme of study
- Number of credits required for each course segment of programme (Eg. Core, Elective, etc.)


```
{
    "SUCCESS": true
}
```


Fig. 6: Sample of service output when genaral request to JWS is made


```
{
  "AWARDTYPE": "FIRSTCLASS",          Class of Degree currently in line for
  "ELECTIVECREDITSREQUIRED": 8,       # elec. credits required for current programme
  "CORECREDITSREQUIRED": 52,          # core credits required to complete current programme
  "NUMORALSAVAILABLE": 2,             # of oral Examinations that student can take
  "ALLOWEDCOURSE": [                  List of courses that student can take based on the courses he / she has taken the past
    "COMP3300",
    "COMP2300",
    "COMP3100",
    "COMP3550",
    "COMP3220",
    "COMP2600",
    "COMP3900"
  ],
  "ELECTIVECREDITSCOMPLETED": 8,      # of elec. credits completed in current programme
  "ADVANCEDCREDITSREQUIRED": 60,      # of adv. credits for current programme (core+elec.)
  "HASALLLEVELONECREDITS": "YES",     has student completed all required lvl. 1 credits?
  "LEVELONECREDITSCOMPLETED": 24,     # of lvl. 1 credits completed in current programme
  "POSSIBLEORALCOURSE": [             List of courses for which a student can possibly apply for an oral examination
    "COMP3100",
    "INFO2420"
  ],
  "LEVELONECREDITSREQUIRED": 24,      # lvl. 1 credits required for current programme
  "HASALLELECTIVECREDITS": "YES",     has student completed all required elec. credits?
  "STUDENTTYPE": "FT",                Type of student {FT : Full Time, PT : Part Time
  "HASALLFOUNDATIONCREDITS": "YES",   has student completed all required foun. credits?
  "YEARSINPROGRAMME": 3,              # of years since the student was enrolled in current programme
```


Fig. 7: Sample of service output when request to retrieve student's advising information is made (Part A)

- Number of credits completed for each course segment of programme (Eg. Core, Elective, etc.)
- List of courses that can be taken by student (based on priority in programme of study)
- Whether or not a student can graduate from current programme of study
- Number of Oral Assessments that the student currently has remaining
- List of all courses which the student can apply for an oral assessment
- Progress in all programmes similar to student's current programme of study

An illustration of this is shown in figures 7 and 8. This information is then consumed by the PWS which then renders it to the user as required.

2) PWS Implementation : The PWS, responsible for controlling all functionality within AdviseMe, provides a wide range of functions and serves three types of users: students, advisors and administrators. A list of the core of AdviseMe is given in the table 2.

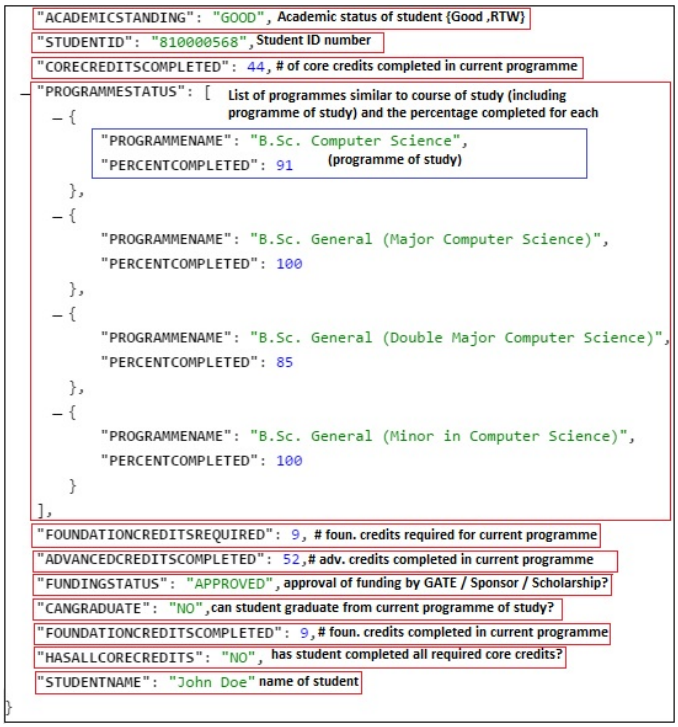


Fig. 8: Sample of service output when request to retrieve student's advising information is made (Part B)

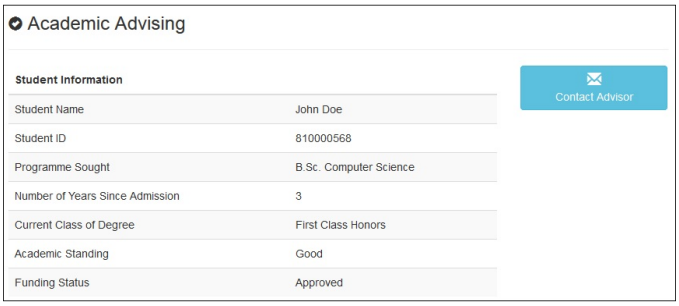


Fig. 9: Sample of Academic Advising Page (Part 1)

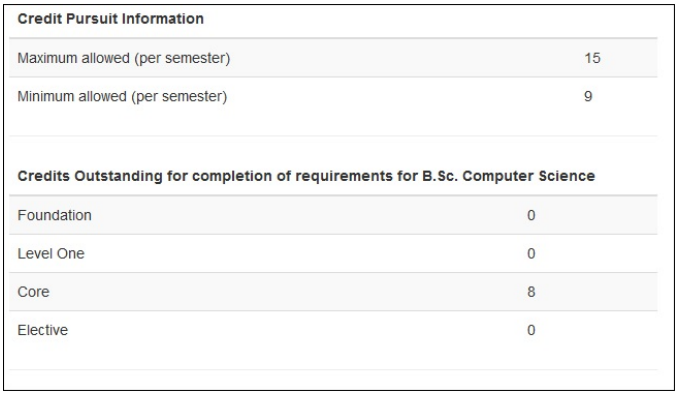


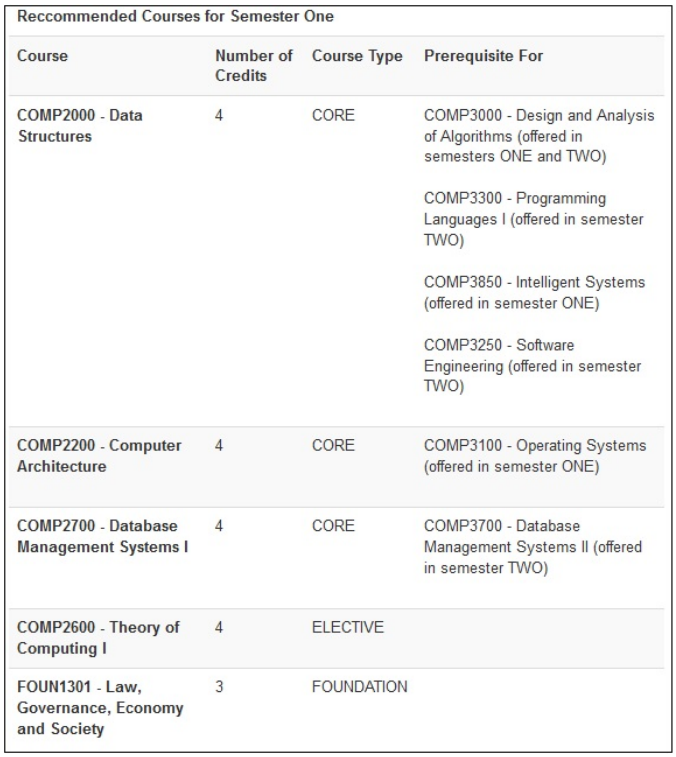
Fig. 10: Sample of Academic Advising Page (Part 2)

TABLE II: Core Features of AdviseMe

Description	Stu	Adv	Adm
View student advising history giving all comments previously placed on student profile by advisors.	YES	-	-
View course advising, by generating a list of courses to be taken per semester as well as show progress in similar programmes in the event of a transfer being considered.	YES	-	-
Request to contact a human advisor in the event that advice given is not sufficient.	YES	-	-
View graduation status of student, showing progress in each of the programme sections.	YES	-	-
View oral examination information, giving the number of orals that a student can still pursue, the number of outstanding credits for completion of programme requirements and the list of courses for which a student can request an oral examination.	YES	-	-
Download a complete summary of all aforementioned information for a student in PDF format.	YES	YES	-
View Information on programme structure as well as common student issues such as requesting overrides and rescindment of RTW status.	YES	YES	-
Place a comment on a students record to be used for future reference by student and advisors.	-	YES	-
Management of all rules within the system inclusive of creation and modification of user defined rules such as prerequisite and exemption rules.	-	-	YES
Management of all courses, programmes, departments, faculties, students' transcripts and users within the context of the university and system.	-	-	YES
Management of the entities associated with the JWS (as mentioned in the previous section).	-	-	YES

As a student, possibly the most useful feature is the "Academic Advising" option, which allows him / her the ability to retrieve information generally sought in academic advising sessions. This initially includes basic student information, his / her academic standing and the class of degree he / she is currently in line for (Figure 9). The proceeding section then gives the maximum and minimum number of credits the student can take per semester as well as the number of credits remaining for each section of the programme (Figure 10). This is then followed by a list of courses (Figure 11) within the current programme of study that can be taken for programme advancement based on semester and also ordered by two levels of priority; the first being how much courses they are prerequisites for and the second being classification by course type (eg. core courses are given higher priority than elective courses). The student is also notified in what semester's possible future courses will be offered so to promote future course planning. Finally, in the last section of the page, the student is able to see his / her progress in similar programmes offered (Figure 12), allowing them to know their stance in other programmes in the event they decide to switch to another programme.

All of these aforementioned sections, in addition to information of graduation status and oral exam assessments give satisfactory advice that answer most common questions faced in academic advising. If however, the student believes that the advice received is insufficient, he / she can then opt to contact a human advisor via email by clicking the "contact advisor" button shown in Figure 9. This then redirects to a new page which allows the student to send an email to the advisor via adviseMe's interface.



Reccommended Courses for Semester One

Course	Number of Credits	Course Type	Prerequisite For
COMP2000 - Data Structures	4	CORE	COMP3000 - Design and Analysis of Algorithms (offered in semesters ONE and TWO) COMP3300 - Programming Languages I (offered in semester TWO) COMP3850 - Intelligent Systems (offered in semester ONE) COMP3250 - Software Engineering (offered in semester TWO)
COMP2200 - Computer Architecture	4	CORE	COMP3100 - Operating Systems (offered in semester ONE)
COMP2700 - Database Management Systems I	4	CORE	COMP3700 - Database Management Systems II (offered in semester TWO)
COMP2600 - Theory of Computing I	4	ELECTIVE	
FOUN1301 - Law, Governance, Economy and Society	3	FOUNDATION	

Fig. 11: Sample of Academic Advising Page (Part 3)

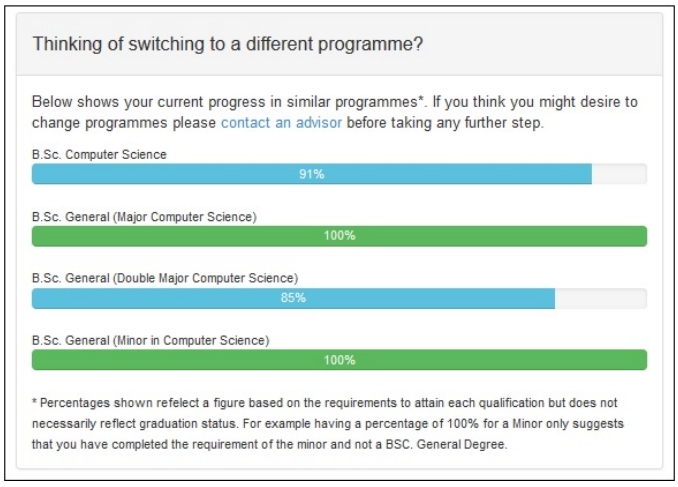


Fig. 12: Sample of Academic Advising Page (Part 4)

C. Post Implementation Phase

To ensure overall acceptance of the system, students of varying aptitudes within the university were allowed to assess the system by either volunteering their own transcript or using a model of a sample transcript. From the 50 percent of students who produced their transcripts for testing, all received accurate advice in all aspects of the system with regard to student advising. In addition to its functional success, all interviewed students found the system to be exceptionally usable and intuitive, commenting that its simplicity and use of visual charts to illustrate student progress made the system very appealing. They also in particular, appreciated the record of advisory history appended to student profiles and the ability to contact a human advisor if necessary. 20 percent of the sample set however, still held strong to the fact that while they would receive sufficient information from the system, they would still opt to speak to an advisor if required. They did appreciate however that the process of meeting an advisor could now be less time consuming as they can now schedule and appointment beforehand, eliminating the need to wait long periods in the advisory offices.

V. Limitations of the System

The system is designed to give accurate advice only to those pursuing "special degrees" or programmes which follow a clear cut path of courses. While this limits the system to serve only a portion of students within FST and other similar institutions, it would still result in the overall reduction of advising load for human advisors, making their duties less demanding.

Another limitation is that while the student can contact a human advisor via email and successfully have their issues solved, the advisor's response time is subjective to when he / she chooses to respond to the student as opposed to face to face conversations where solutions are immediately discussed. This can be unsatisfactory to students, especially if advisors take too long to respond to student concerns.

The system is also in the prototype phase and hence requires administrators to input the student transcripts for system processing via the user interface. While UI design makes the uploading process possible in minimal time, it still results in an unnecessary action by a human entity. However, this timeframe is relatively short, which can be deemed quite acceptable as the number of hours saved by both students and advisors using the system would clearly offset the setup period.

VI. Future Work

With respect to the limitations discussed above, further enhancements are proposed to investigate ways in which the system would be able to sufficiently accommodate students pursuing general degrees. Also in terms of data integration, measures to have the system automatically process information from the Banner Student Information System will be explored, in an attempt to reduce administration involvement further.

With respect to added system functionality, measures would be put in place to offer a higher level of student advisor communication, possibly including live chats with advisors via Instant Messaging or Video Conferencing. Also the idea of a blog whereby a community handling frequently asked questions or common issues would be explored. This would possibly increase the level of acceptance of the system by students and solve the issue of delayed responses that can be incurred while waiting for advisors to attend to emails.

Finally, with respect to future long term enhancements of AdviseMe, a programme planning module can be implemented, using collected data from student advising to generate statistics for use by departmental administration to assist in the allocation of teaching resources for upcoming semesters. This would not only add to the value of the system, but increase its scope of alleviating manual processing issues within tertiary education institutions.

VII. Conclusion

Utilizing AdviseMe in order to facilitate academic advising without the possible involvement of human advisors will definitely enhance the efficiency, integrity and transparency of any tertiary based advising process similar to that of FST. The successful prototype discussed in this paper highlights its feasibility and practicality in the context of UWI and their degree programmes. Serving a significant portion of the student body, it would provide sufficient advisory services to the majority of its users thus reducing the student load faced by human advisors at advisory offices. For those students whose issues go beyond the scope of its assistance, it allows an avenue of communication to human advisors that was previously non-existent, by means of email technology. Not only does this create a flexible way of seeking advice, but it can also improve the quality of the qualitative advice received from advisors, since more time can now be dedicated to handling these special cases.

Its current architecture supports the use of a PHP based web application interacting with an intelligent, RESTful Java Web Server, in order to provide expert advice on course scheduling issues via any device that supports internet browsing. However, the fact that the intelligence processing is exposed via web services indicates that the system can be adapted to suit any future front end deployment, provided that access to the JWS API is given. This promotes acceptance of AdviseMe, outside of UWI, as solutions can be tailored to any university context, once their programme structure is the similar to that of UWI. Returning to the context of FST, we see that a significant number of students showed possible acceptance of the system in the infant stages of conceptualization. This figure sought to increase nearing the end of implementation however as all students interviewed for testing made positive remarks about the system's functionality, usability and applicability to the advising context of FST; with the majority of them stating that they would use such a system. Furthermore, when demonstrated to the Head of the Advisory Unit of FST, she was highly pleased with the outcome, to the point of suggesting possible practical implementation within the faculty in the near future.

With such positive feedback, and the fact that the resulting product captured all the requirements that was proposed of such a system, it is clear that the implementation of AdviseMe was indeed a success, lending its services to support new and existing advising processes in order to enhance the overall quality of academic advising received by students today.

References

[1] E. Nwelih and S. Chiemeke, "Framework for a web-based spatial decision supportsystem for academic advising," *African Journal of Computing and ICT*, vol. 5, no. 4, pp. 121–126, 2012.

[2] M. T. Al-Nory, "Simple decision support tool for university academic advising," in *Information Technology in Medicine and Education (ITME), 2012 International Symposium on*, vol. 1. IEEE, 2012, pp. 53–57.

[3] F. Lin, S. Leung, D. Wen, F. Zhang, and M. Kinshuk, "e-advisor: A multi-agent system for academic advising," *International Transactions on Systems Science and Applications*, vol. 4, no. 2, pp. 89–98, 2008.

[4] F. Albalooshi and S. Shatnawi, "Online academic advising support," in *Technological Developments in Networking, Education and Automation*. Springer, 2010, pp. 25–29.

[5] T. Feghali, I. Zbib, and S. Hallal, "A web-based decision support tool for academic advising," *Journal of Educational Technology & Society*, vol. 14, no. 1, pp. 82–94, 2011.

[6] M. A. Al Ahmar, "A prototype student advising expert system supported with an object-oriented database," *International Journal of Advanced Computer Science and Application*, vol. 1, no. 3, pp. 100–105, 2011.

[7] M. Beheshti, T. Trang, K. Kowalski, and J. Han, "Student advising system," in *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, vol. 2006, no. 1, 2006, pp. 2727–2732.

[8] O. Daramola, O. Emebo, I. Afolabi, and C. Ayo, "Implementation of an intelligent course advisory expert system," *International Journal of Advanced Research in Artificial Intelligence*, vol. 3, no. 5, pp. 6–12, 2014.

[9] L. Mostafa, G. Oately, N. Khalifa, and W. Rabie, "A case based reasoning system for academic advising in egyptian educational institutions."

[10] A. N. Nambiar and A. K. Dutta, "Expert system for student advising using jess," in *Educational and Information Technology (ICEIT), 2010 International Conference on*, vol. 1. IEEE, 2010, pp. V1–312–V1–315.

[11] K. Kowalski, J. Goetz, and M. Alam, "Intelligent on-line advising with expert system shell," in *World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, vol. 2006, no. 1, 2006, pp. 687–694.

[12] M. J. Martínez-Argüelles, E. Ruiz-Dotras, and E. Rimbau-Gilabert, "The academic advising system in a virtual university," in *Technology Enhanced Learning. Quality of Teaching and Educational Reform*. Springer, 2010, pp. 345–350.

[13] A. Al-Ghamdi, S. Al-Ghuribi, A. Fadel, and F. AL-Ruhaili, "An expert system for advising postgraduate students," *International Journal of Computer Science and Information Technologies*, vol. 3, no. 3, 2012.