Software Engineering
Software Requirements Specification (SRS) Document

Matthew Seiler
Danielle Paredes
Forrest Meade
Isaac McCraw
Nathan Velasquez

Bowtie Code

www.radford.edu/softeng10

March 27th, 2014

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft 0.1	Matthew Seiler, Forrest Meade	Partial draft including introductory and functional requirements sections	03/25/14
Draft 0.2	Matthew Seiler, Forrest Meade, Nathan Velasquez, Danielle Paredes, Isaac McCraw	Pasted in references, performance requirements, and usage scenarios. Added interface requirements, UML use case diagram, non-functional requirements	03/27/14

Review & Approval

Requirements Document Approval History

Approving Party	Version Approved	Signature	Date
Project Manager			
Dr. T. L. Lewis			

Requirements Document Review History

Reviewer	Version Reviewed	Signature	Date
Danielle Paredes	Draft 0.2	*Danielle Paredes*	3/27/14
Forrest Meade	Draft 0.2	*Forrest Meade*	3/27/14
Isaac McCraw	Draft 0.2	*Isaac McCraw*	3/27/14
Matthew Seiler	Draft 0.2	*Matthew Seiler*	3/27/14
Nathan Velasquez	Draft 0.2	*Nathan Velasquez*	3/27/14

Contents

1. Introduction 3
   1.1 Purpose of this document 3
   1.2 Scope of this document 3
   1.3 Overview 3
   1.4 Business Context 3
2. General Description 4
   2.1 Product Functions 4
   2.2 Similar System Information 4
   2.3 User Characteristics 4
   2.4 User Problem Statement 4
   2.5 User Objectives 5
   2.6 General Constraints 5
3. Functional Requirements 5
4. Interface Requirements 11
   4.1 User Interfaces 11
   4.2 Hardware Interfaces 13
   4.3 Communications Interfaces 13
   4.4 Software Interfaces 13
5. Performance Requirements 13
6. Other non-functional attributes 13
   6.1 Security 13
   6.3 Reliability 13
   6.4 Maintainability 13
   6.5 Portability 14
   6.6 Extensibility 14
   6.7 Reusability 14
   6.8 Application Affinity/Compatibility 14
7. Operational Scenarios 14
8. Preliminary Use Case Models and Sequence Diagrams 15
   8.1 Use Case Model 15
   8.2 Sequence Diagrams 15
9. Updated Schedule 15
10. Appendices 15
   10.1 Definitions, Acronyms, Abbreviations 15
   10.2 References 16

1. Introduction

1.1 Purpose of this document

The purpose of this document is to provide a detailed description about the requirements needed to successfully complete the Radford University Agenda mobile application for the benefit of the project management, client, and development team. Throughout this document, Bowtie Code will provide a general description of the project, the functional, interface, and performance requirements for the application, a list of other relevant application attributes, application usage scenarios, and use case diagrams, as well as an updated schedule

1.2 Scope of this document

This requirements effort intends to define the requirements for the RU Agenda mobile application. The requirements elicitation team includes Danielle Paredes, Forrest Meade, Isaac McCraw, Matthew Seiler, Nathan Velasquez, and the client, Mr. Ryan Skipper. The team will spend the period between March 4, 2014 and March 27, 2014 identifying and documenting functional, interface, performance, and other requirements for the RU Agenda software system. Actual time spent on requirements elicitation during that period will be considerably less than the total 23 days scheduled, as the Spring Break holiday and midterm exams are included in the same time span.

1.3 Overview

The RU Agenda mobile application is intended to provide Radford University students with a way to track their class assignments and due dates using their mobile device. The application will maintain a list of assignments which can be viewed with all assignments in order by due date or with assignments grouped by class. Each assignment will include a title, description, class, and due date. The user can add assignments to the list, modify existing assignments, or select one or more assignments to remove from the list. Additionally, the app can optionally notify the user a set time before the due date of an assignment, automatically remove past-due assignments, and automatically download the user's class schedule from Radford University's systems.

1.4 Business Context

The development of this software system is sponsored by the Software Engineering program in the Department of Information Technology at Radford University. The Software Engineering concentration is intended to produce designers and developers for large, complex software systems. The program introduces students to the concepts and techniques required to build large software systems, and provide opportunities to obtain practical experience applying those techniques on an actual development effort.

2. General Description

2.1 Product Functions

The RU Agenda app will allow Radford University students to access their class schedules and manage their assignment information through an Android application. The finished application will provide a way for users to access their class schedule by logging in to Radford University's systems, add, remove, and edit assignment info in a list of assignments, and share assignment information with other users

2.2 Similar System Information

myHomework is a to do app that provides a similar task organization. It is similar in ways such as listing the current classes you are in, and allowing you to view your assignments by class and by priority. One advantage that *myHomework* has is that it allows you to select the type of assignment. On the other hand, our app will be able to alert users to assignment due dates and import the user's class schedule automatically, which *myHomework* is not capable of.

Koofers is an online database of class information for universities nationwide, which allows users to build a class schedule and access class information such as assignments and exams from past semesters. It has many more features than the proposed RU Agenda application, such as a GPA calculator and professor ratings. However, the scope of the website is much larger than the scope of the proposed project; the RU Agenda aims to do few things but do them well and in a way that will be most convenient for student users at Radford University.

The *Apple iOS Reminders* app allows users to associate a list of tasks with an email account, and create tasks and reminders under that account that can notify the user near planned completion time. The app boasts quick and seamless integration with iOS on the iPhone, but is not targeted to student users, and thus is not class-based, cannot import a user's class schedule, and does not provide task attributes related to class assignments.

2.3 User Characteristics

The RU Agenda application is intended to be used by Radford University students. A typical user should have some experience using web applications and mobile operating systems. A basic knowledge of the iOS or Android operating system on a mobile device as well as the ability to authenticate to Radford University's online student services will be sufficient to use the application effectively.

2.4 User Problem Statement

Although there exist many applications designed to help users manage their schedules and tasks, there are currently no solutions which are designed specifically for Radford University students. Other applications require that users manually manage their task categories and task attributes, whereas an application which is targeted at students and is integrated with Radford University's student services would allow significantly more automation and greater ease of use.

2.5 User Objectives

After signing in with my Radford University username and password, the application should automatically import my current class schedule and any assignments associated with those classes which have been uploaded by my instructors. The application should allow me to add new assignments, edit current assignments, and remove old assignments. The applications should alert me if there is an assignment or test due soon, and should make it easy to quickly check what assignments are due next and which assignments are due for a particular class. I should also be able to share the information about an assignment with another student.

2.6 General Constraints

The application must run on the Android mobile operating system. A user's Android device must provide network connectivity (both hardware and software) in order for the application to fully function. The system must use class data and authentication systems which are already in place and provided by Radford University.

3. Functional Requirements

Criticality Scale

Non-Critical	Non
Less Critical	Less
Critical	Normal
Very Critical	Very
Extremely Critical	Extreme

1. **The application shall maintain a list of classes for the user**
    a. **Description** - The application will store a list of academic class information for the user. This list should be accessible, editable, and persistent between application lifetimes. The associated attributes of a class are described in subsequent additional requirements.
    b. **Criticality** - Extreme
    c. **Technical issues** - The app must use input provided by the user and/or retrieved from an external web service to create a list of classes associated with the current user. The app should also be able to store this list in such a way that it is persistent if the application is terminated and restarted.
    d. **Risks** - Low risk - technical implementation is relatively straightforward and there is no dependency on network or remote database connectivity.
    e. **Dependencies** - none
2. **A class shall include the course title**
    a. **Description** - One attribute of a class as stored by the application will be the title of the course, for example "Web Programming II" or "Software Engineering I".
    b. **Criticality** - Very
    c. **Technical issues** - This attribute must be generated from either direct user input or data retrieved from a web service or remote database.

    d. **Risks** - Course titles can be verbose - if an attribute must be dropped because of screen real-estate concerns on the user input form for class attributes, this one will go first.
    e. **Dependencies** - 1.
3. **A class shall include the course number**
    a. **Description** - Another attribute of a class is the course number for that class, for example “ITEC 320-01” or “ITEC 345-02”.
    b. **Criticality** - Very
    c. **Technical issues** - This attribute must be generated from either direct user input or data retrieved from a web service or remote database.
    d. **Risks** - Low risk - implementation is simple and does not require complex interactions with other requirements
    e. **Dependencies** - 1.
4. **A class shall include the location where the class meets**
    a. **Description** - Another attribute of a class is the location where the class meets, expressed in terms of a campus building and room number, like “Davis 114” or “Young 123”.
    b. **Criticality** - Very
    c. **Technical issues** - This attribute must be generated from either direct user input or data retrieved from a web service or remote database.
    d. **Risks** - Low risk - implementation is simple and does not require complex interactions with other requirements
    e. **Dependencies** - 1.
5. **A class shall include the days and times when the class meets**
    a. **Description** - Another attribute of a class are the days of the week and times of the day when that class meets, for example “M W F 10:00am - 11:50am”.
    b. **Criticality** - Very
    c. **Technical issues** - This attribute must be generated from either direct user input or data retrieved from a web service or remote database.
    d. **Risks** - Non-plaintext data could be more complicated to collect, store, and output appropriately.
    e. **Dependencies** - 1.
6. **A class shall include the name of the instructor who teaches the class**
    a. **Description** - Another attribute of a class is the name of the instructor who teaches the class, for example “Tracy Lewis-Williams” or “Edward G. Okie”.
    b. **Criticality** - Normal
    c. **Technical issues** - This attribute must be generated from either direct user input or data retrieved from a web service or remote database.
    d. **Risks** - It is highly unlikely that this attribute would need to be removed, but it is lower priority than the What (3), Where (4), and When (5) attributes and could be removed if screen space is heavily constrained.
    e. **Dependencies** - 1.
7. **Users shall be able to add a class to a list of classes in which they are enrolled.**
    a. **Description** - When using the application, any user should be able to add a class and its corresponding details to the list of classes the application maintains.
    b. **Criticality** - Very

c. **Technical issues** - When the user presses the associated button in the application, the application should present the user with a form to allow them to add class details. After adding class details, the app should create a class instance with data fields corresponding to the user input and add the instance the list of the user's classes.
d. **Risks** - Low risk - technical implementation is relatively simple and there no app-external dependencies (no network and/or remote database connectivity is required).
e. **Dependencies** - 1.

8. **Users shall be able to remove classes from the list of classes in which they are enrolled**
    a. **Description** - When using the application, any user should be able to remove a particular class from their list of classes.
    b. **Criticality** - Very
    c. **Technical issues** - The application should be able to remove the class from the list of classes which it maintains locally. This change should be reflected into persistent local storage, but not to any external web service or remote database.
    d. **Risks** - Low risk - implementation is simple and does not require complex interactions with other requirements
    e. **Dependencies** - 1.
9. **Users shall be able to completely reset their lists of classes and assignments**
    a. **Description** - This function would allow the user to reset their agenda to an empty state in between semesters by permanently removing all classes and all assignments associated with those classes.
    b. **Criticality** - Less
    c. **Technical issues** - The significant data loss associated with this function necessitates a confirmation prompt at minimum, or perhaps even a temporary undo capability, but undo capability would drastically increase the complexity of the function.
    d. **Risks** - Since removing a single class at a time would allow the user to clear all of their classes and achieve an identical state (simply in a less convenient manner) this requirement receives a lower priority and could be dropped if development time is constrained.
    e. **Dependencies** - 1.
10. **Users shall be able to authenticate to the application using their Radford University username and password**
    a. **Description** - The system should be able to verify the Radford University credentials of a user in order to authenticate the user to the university's web services.
    b. **Criticality** - Non
    c. **Technical issues** - Precondition: satisfaction of this requirement depends on the capability to leverage Radford University's authentication system.
    d. **Risks** - Radford University's authentication systems are obfuscated from the development team and may not be accessible in any way.
    e. **Dependencies** - External
11. **The application shall be able to automatically download the class schedule of**

authenticated users

a. **Description** - The application should be able to automatically import the class schedule of an authenticated user (see req. 10) for the current semester on request. Accessing the user's class data using a web service or remote database would allow the application to populate the user's list of classes without taking direct text input from the user.
b. **Criticality** - Non
c. **Technical issues** - Precondition: this functionality depends on the availability of appropriate external web services and network connectivity.
d. **Risks** - Depends entirely on 10 and the general capability to access web services associated with Radford University. The capability is also complex enough that the team may not be able to implement it in the time available for the project.
e. **Dependencies** - 1, 10.

12. **The user shall be able to view their list of classes**
    a. **Description** - The application should provide a list view for the user of all the classes currently stored by the app.
    b. **Criticality** - Extreme
    c. **Technical issues** - Precondition: App stores a list of class/course instances; Post condition: graphical presentation of the classes to the user in list format
    d. **Risks** - Low risk - simple implementation and no complicated dependencies.
    e. **Dependencies** - 1. (list must exist to be viewed)
13. **The app shall maintain a list of assignments associated with each class in the list of classes**
    a. **Description** - Every class in the list of classes will have another list associated with it to store assignments related to that class.
    b. **Criticality** - Extreme
    c. **Technical Issues** - Using a user's list of classes and additional direct user input, the app can create and store a list of assignments for each class.
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 1. (class list must exist to associate assignments with classes)
14. **Users shall be able to add an assignment to a list of assignments**
    a. **Description** - The application should provide the capability for the user to add a new assignment to the list of assignments for a class.
    b. **Criticality** - Very
    c. **Technical issues** - Input: Existing class, direct user input. Output: Assignment instance added to list of assignments for class
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 13.
15. **Users shall be able to edit assignments in their list of assignments**
    a. **Description** - The application should provide the capability for the user to edit the attributes of an existing assignment in the list of assignments for a particular class.
    b. **Criticality** - Normal
    c. **Technical issues** - Input: Existing assignment, direct user input. Output: Modified assignment instance in list of assignments
    d. **Risks** - Low risk - simple implementation and no complex dependencies

e. **Dependencies** - 13.

16. **Users shall be able to remove an assignment from their list of assignment**
    a. **Description** - The application should provide the capability for the user to remove an assignment from the list of assignments for a particular class.
    b. **Criticality** - Very
    c. **Technical issues** - Input: List of assignments, user chooses an assignment to remove Output: Chosen assignment permanently removed from list of assignments
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 13.
17. **Users shall be able to clear all assignments for a particular class**
    a. **Description** - The application should provide the capability for the user to remove all assignments from the list of assignments for a particular class with a single action.
    b. **Criticality** - Normal
    c. **Technical issues** - Input: existing class, direct user interaction. Output: All assignments for selected class permanently removed
    d. **Risks** - Since removing all assignments can be accomplished by removing assignments individually, this requirement is lower priority than requirement 16 and might not be completed if development time is constrained
    e. **Dependencies** - 13.
18. **The user shall be able to set the name of an assignment**
    a. **Description** - One attribute of an assignment should be a name or title for the assignment (less than 255 characters in length)
    b. **Criticality** - Normal
    c. **Technical issues** - Input: new or existing assignment, direct user input (text). Output: new name attribute of assignment instance
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 13, 14, 15.
19. **The user shall be able to set the description of an assignment**
    a. **Description** - Another attribute of an assignment should be an optional detailed description of the assignment (roughly 0 to 1000 characters)
    b. **Criticality** - Normal
    c. **Technical issues** - Input: new or existing assignment, direct user input (text). Output: new description attribute of assignment instance
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 13, 14, 15.
20. **The user shall be able to set the due date of an assignment**
    a. **Description** - Another attribute of an assignment should be the date and time when the assignment is due
    b. **Criticality** - Normal
    c. **Technical issues** - Input: new or existing assignment, direct user interaction. Output: new due date attribute of an assignment instance
    d. **Risks** - Time and date parsing and calculation is reasonable complicated, and the team might not be able to fulfill this requirement in a satisfactory manner if unable to use a third-party date picker UI component

    e. **Dependencies** - 13, 14, 15.

21. **The user shall be able to set the class which an assignment belongs to**
    a. **Description** - When creating an assignment or when editing an assignment, the user should be able to specify which class from their list of classes the current assignment is associated with
    b. **Criticality** - Very
    c. **Technical issues** - Input: new or existing assignment, direct user interaction, existing class. Output: sets association between class and assignment
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 1, 13, 14, 15.
22. **The user shall be able to set notification parameters for an assignment**
    a. **Description** - The user should be able to optionally specify a length of time before the due date of an assignment
    b. **Criticality** - Less
    c. **Technical issues** - Input: new or existing assignment, direct user input. Output: new notification settings for an assignment
    d. **Risks** - The settings interface should be relatively easy to implement, however the overall notifications system (req. #26) may be more complex. If the related requirement is not completed then this feature will be stripped for consistency.
    e. **Dependencies** - 13, 14, 15, 26.
23. **The user shall be able to view their list of assignments ordered by due date**
    a. **Description** - The application should provide the user with a way to view all of their assignments for all of their classes together, ordered by due date.
    b. **Criticality** - Very
    c. **Technical issues** - Input: Lists of assignments for every class. Output: Combined list sorted by due date.
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 13.
24. **The user shall be able to view their list of assignments grouped by class**
    a. **Description** - The application should provide the user with a way to view all of the assignments for only one class, ordered by due date
    b. **Criticality** - Normal
    c. **Technical issues** - Input: Existing list of assignments for each class. Output: presentation of the same
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 1, 13.
25. **The user shall be able to view the detailed attributes of a single assignment**
    a. **Description** - The application should be able to present the user with all of the attributes of one assignment in a single view
    b. **Criticality** - Normal
    c. **Technical issues** - Input: User-selected existing assignment. Output: presentation of all attributes of the selected assignment
    d. **Risks** - Low risk - simple implementation and no complex dependencies
    e. **Dependencies** - 13.
26. **The application shall be able to notify the user when an assignment is close to its due date**

a. **Description** - If the application should be able to present the user with a notification at the appropriate time if the user has specified that option for a particular assignment.
b. **Criticality** - Less
c. **Technical issues** - Input: Notification attribute of existing assignment. Output: UI notification of approaching due date.
d. **Risks** - If the application framework does not provide a mechanism for triggering UI notifications, then this requirement may not be able to be completed. Additionally, if implementation of this feature is found to be overly complex then it may not be completed in the development time allotted for the project.
e. **Dependencies** - 13, 22.

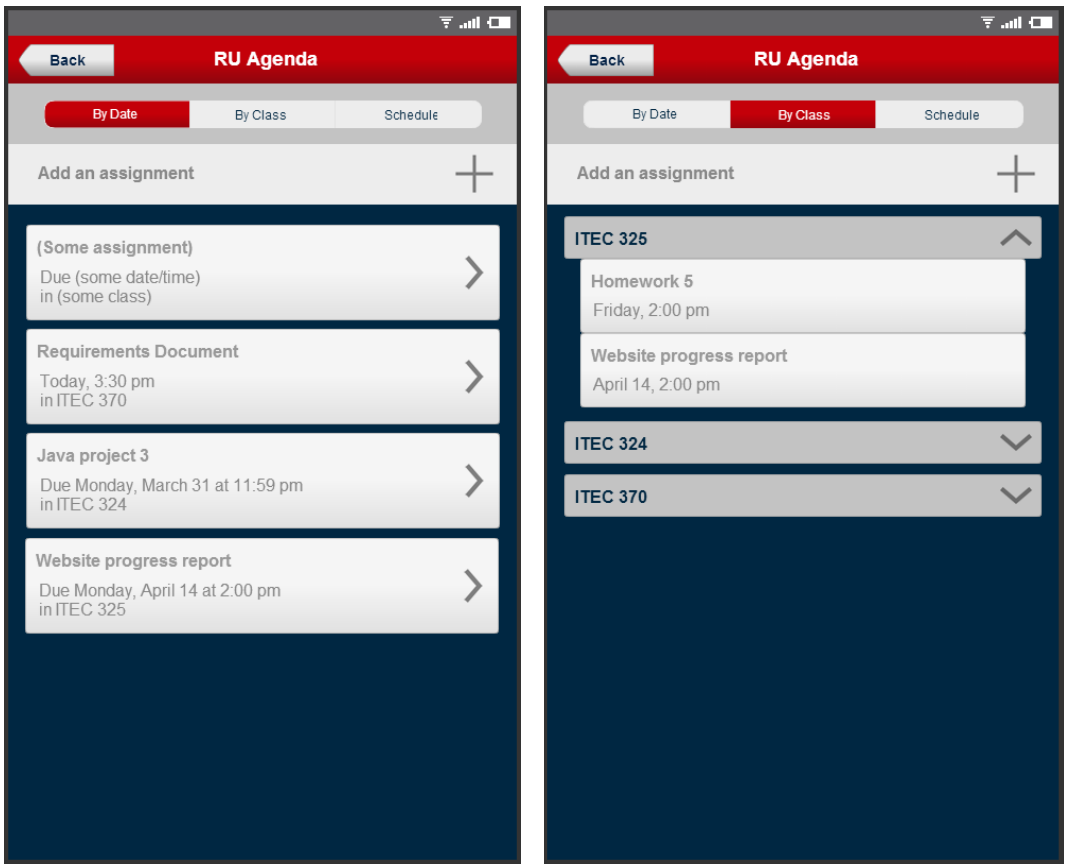
4. Interface Requirements

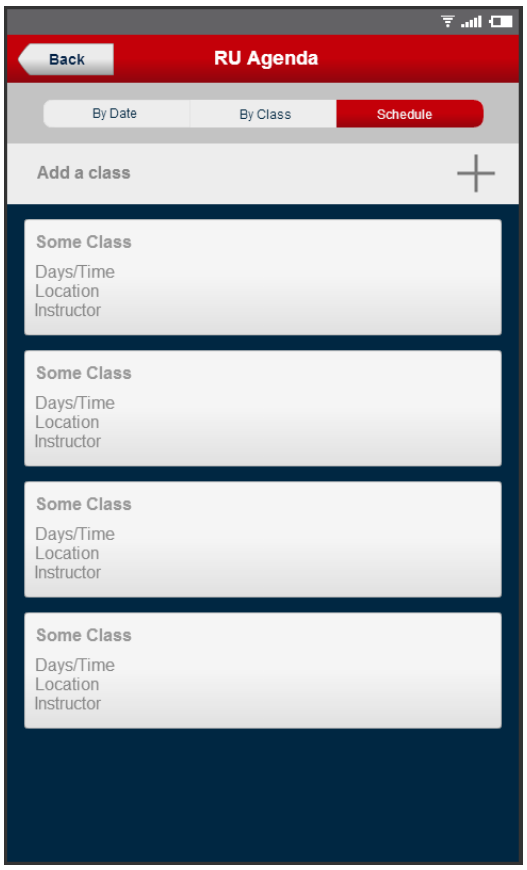
4.1 User Interfaces

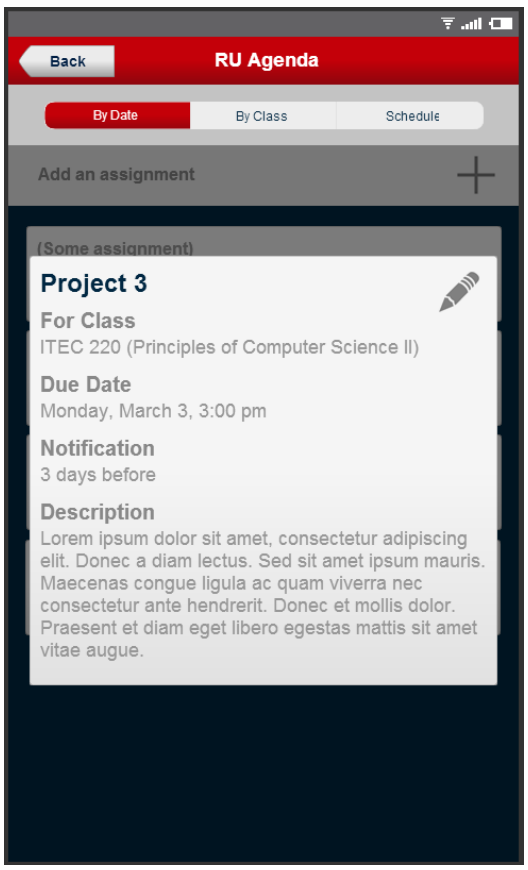
The interface shows the user notifications for meetings in a personally designed agenda and allows the user to continually customize their schedule by allowing the user to add events and classes by a simple touch of the screen. Then the user may customize that event to specific detail and specification.

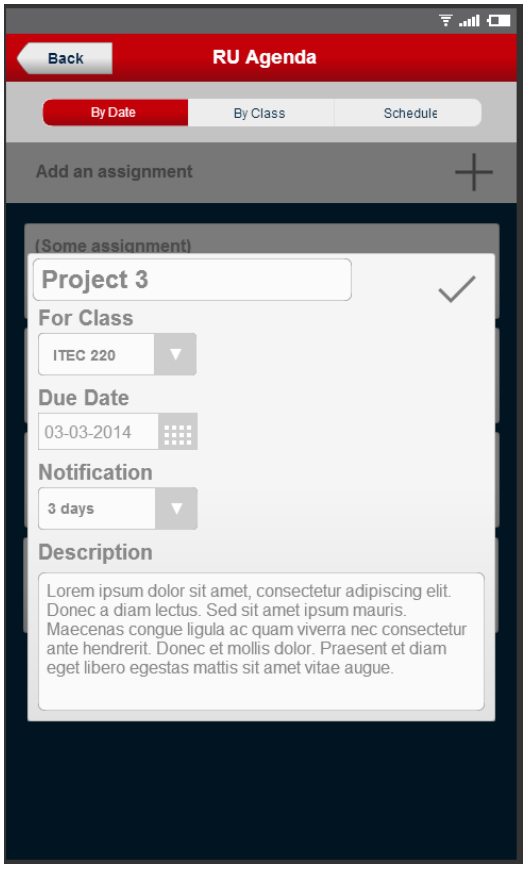
4.1.1 GUI

Screen Mockups:




Back
RU Agenda
By Date
By Class
Schedule
Add a class
Some Class
Days/Time
Location
Instructor
Some Class
Days/Time
Location
Instructor
Some Class
Days/Time
Location
Instructor
Some Class
Days/Time
Location
Instructor


Back
RU Agenda
By Date
By Class
Schedule
Add an assignment
(Some assignment)
Project 3
For Class
ITEC 220 (Principles of Computer Science II)
Due Date
Monday, March 3, 3:00 pm
Notification
3 days before
Description
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a diam lectus. Sed sit amet ipsum mauris. Maecenas congue ligula ac quam viverra nec consectetur ante hendrerit. Donec et mollis dolor. Praesent et diam eget libero egestas mattis sit amet vitae augue.


Back
RU Agenda
By Date
By Class
Schedule
Add an assignment
(Some assignment)
Project 3
For Class
ITEC 220
Due Date
03-03-2014
Notification
3 days
Description
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec a diam lectus. Sed sit amet ipsum mauris. Maecenas congue ligula ac quam viverra nec consectetur ante hendrerit. Donec et mollis dolor. Praesent et diam eget libero egestas mattis sit amet vitae augue.

4.1.2 CLI

N/A

4.2 Hardware Interfaces
The app is designed for the Android phone and market.

4.3 Communications Interfaces
The application will utilize the networking hardware of the user's device through network communications services provided by the Android operating system. Network communications capability will be used to connect to remote databases and/or web services for authentication and importing the user's class schedule information.

4.4 Software Interfaces
None additional

5. Performance Requirements

System
The application will run on all Android devices running 4.1 (JellyBean) or Later. It will be around 11mb in size. The application will respond to the size of the screen and/or window the application is running in.

Response Time
The application should take less than 4 seconds when running on an Android phone and less than 8 second when on an emulator or tablet. The application will run fine until the user begins to multi-task between 3 or more processes.

Workload
The application must support approximately 10,000 users at the time of launch based on the population of the RU student body.

Scalability
The application will be able to scale to the size of the RU student body as it increases.

6. Other non-functional attributes

6.1 Security
Users must be a Radford University student in order to access the automatic class schedule and assignment sharing features of this application Users can authenticate by logging in using their RU username and password.

6.3 Reliability
Most functionality will not require network connectivity. System components that require authentication through Radford University and network connectivity will function as long as the systems maintained by the University are available.

6.4 Maintainability
The development team will follow best practices for clean code and software modularity in order to make the application as maintainable as possible.

6.5 Portability

Users will be able to access this application on or off campus anytime on their mobile device without the need of an internet connection.

6.6 Extensibility

The application will be highly extensible in terms of adding course and calendar details or views. However, the application in general has low extensibility.

6.7 Reusability

An application instance shall be able to be reusable every semester for a returning student. The user will be able to update their class schedule by having the application check for a registered schedule set for the next semester and if the current school semester is over. Application components will be able to be integrated into the larger RU Mobile application developed and maintained by Radford University.

6.8 Application Affinity/Compatibility

The application shall be compatible with Android 4.1 or any later version.

7. Operational Scenarios

1) Student arrives on campus the day before the semester starts and is informed about the RU Agenda phone app by a friend. The student then begins to download and install the application on to his/her mobile device. The student logs in to the application; having already registered for classes; the application downloads the student's schedule for them.

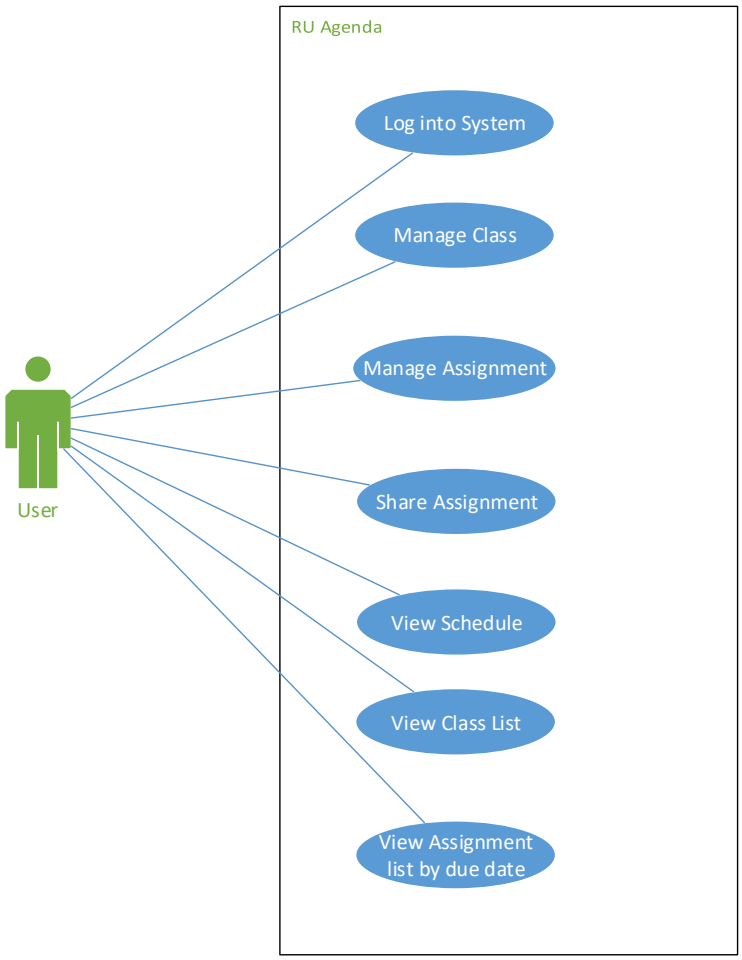
2) The semester begins and a new student has recently downloaded the new Radford University app called RU Agenda. The student is on his way to his to his first class but before he leaves he checks his RU Agenda app that has automatically downloaded his class schedule from a Radford database to make sure he is headed to the correct class at the correct time. He logs in using his new RU login information and notices his class is exactly when he thought and also notices while maneuvering through the app that he has an assignment due in the class already. Conveniently he acquires details about the assignment from the app and realizes the assignment is not due for a few weeks. He carries on to his class as scheduled.

3) During the semester, the student is in one of his/her classes and the professor assigns an assignment to the class. The student then goes onto RU Agenda to that specific class. There he/she adds the assignment with its title, description, and due date for the assignment. Then it is added to the list of assignments for that class conveniently for the student. The student then logs off and continues listening in class.

4) While working on a homework assignment, the student receives an email from their instructor notifying them that the due date for the assignment has been postponed. The student had already added this assignment in their RU Agenda. They navigate to the view of assignments grouped by class, select the appropriate class, and select the assignment to open a detailed view. From there they edit the due date for the assignment to the new, postponed due date, and check the notification settings for the assignment. The next day, after completing the assignment, they open the RU Agenda app and remove the assignment from their list.

8. Preliminary Use Case Models and Sequence Diagrams

8.1 Use Case Model



8.2 Sequence Diagrams

(Omitted by Instructor)

9. Updated Schedule

(See attached Excel document Project-Schedule.xlsx)

10. Appendices

10.1 Definitions, Acronyms, Abbreviations

"RU" – Radford University
"App" – Application

10.2 References

"myHomework Student Planner App." instin, LLC., 2014. Web. 27 March 2014. < https://myhomeworkapp.com/ >

"Koofers." Koofers, Inc., 2014. Web. 27 March 2014. < https://www.koofers.com/ >

"iOS: Using Reminders." Apple, Inc., 2014. Web. 27 March 2014. < https://support.apple.com/kb/HT4970 >

"Software Engineering." Dept. of Information Technology at Radford University, 2014. Web. 27 March 2014. < http://www.radford.edu/content/csat/home/itec/programs/computer-science/software-engineering.html >

Lewis-Williams, Tracy. "ITEC 370 - Software Engineering I - Course Syllabus" 2014. PDF file.