Nicholas Guida

Steven Lu

RUPLANNER

## Overview

The goal of this project is to provide an intuitive, easy to use planner with which students can “plan” out their entire college career with. This planner will provide powerful functionality and will indicate:

- How many credits you currently have
- How many more credits you need to graduate
- Which classes require prerequisite classes
- Which classes count towards your major
- Requirements needed for graduation
- Difficulty of course

As well as many other useful benefits like saving, editing, and sharing your planner with others. The main importance of this project is to make it as friendly as possible by providing all necessary information in an elegant manner.

## Customers

This planner, while available to whomever wants to register for it, will only pertain to Rutgers engineering students. While a course list will exist with all available courses, the options for major will only include the school of engineering majors. This is purely due to time constraints since each major has different specifications and required courses needed to be taken for graduation.

## System Functions

### Evident

- GUI with which users interact with to create/edit/delete/save their planner
- List of all available courses
- Log in/register screen
- Dynamically updated information regarding credits, prerequisites, course difficulty, etc
- List of available engineering majors to select from
- Share button for emailing planner/social media sharing

### Hidden

- Planner information being saved in the database
- Login / registration information being saved
- Calculations and algorithms for determining total credits and prerequisites

### Frill

- Color indicators for unavailable/available courses
- Warning about level of difficulty for classes (i.e very easy/very hard)

## System Attributes

### Details

- Very clean and elegant GUI for users to use
- planners/login information stored in SQL database
- HTML/CSS/PHP for web page layout
- Javascript implementation for the dynamically updating counters and requirements
- Rutgers' APIs used for course list and integration
- Algorithm for suggesting certain courses to users

### Constraints

- Planner will only provide full functionality to engineering majors
- Doesn't have list of time schedule for courses, so planned courses may time conflict in reality
- Graduation requirements change frequently, so updating backend information will be required periodically

### Must

- Provide an accurate course list
- Provide an accurate requirement list and credit count
- Show required prerequisites and course conflicts
- Accurately show remaining classes needed to be taken
- Allow for the creation, editing, deletion, and saving of planners
- Allow for registration and easy log in
- Work with all engineering majors

**Want**

- Sharing of planners via email or social media
- List difficulty of course based off of past data
- Recommend courses to be taken based off of current information
- Add a “minor” selection which adds further requirements for minor

## Use Cases

**Requirements:**

| Number | Requirement |
|---|---|
| REQ1 | The system shall provide a template for students to create a schedule of classes for their entire college career. |
| REQ2 | The system shall allow the student to register and login to the website in order to provide viewing, saving and editing functions for planners. |
| REQ3 | The system shall allow the editing, deleting, and saving of planners while also allowing for the adding and removing of classes for each planner. |
| REQ4 | The system shall provide a clean and aesthetically pleasing UI which clearly indicates any and all information being provided to the user. |
| REQ5 | The system shall provide the student valuable information regarding important including, but not limited to, prerequisite courses, total credits taken, required courses needed to be taken, and possible replacement courses for required credits. |
| REQ6 | The system shall have a way to share a user’s course planner with others via easy social media integration and/or email exporting. |
| REQ7 | The system shall show all available courses for the semester including the difficulty of the course, so as to provide a balanced planning approach for students who don’t want too many hard (or too many easy) courses. |
| REQ8 | The system shall provide a list of “back up” courses which students can add too in case their originally planned courses overlap or become full during registration |
| REQ9 | Allows the Admin to update the requirements for the majors due to university changes |

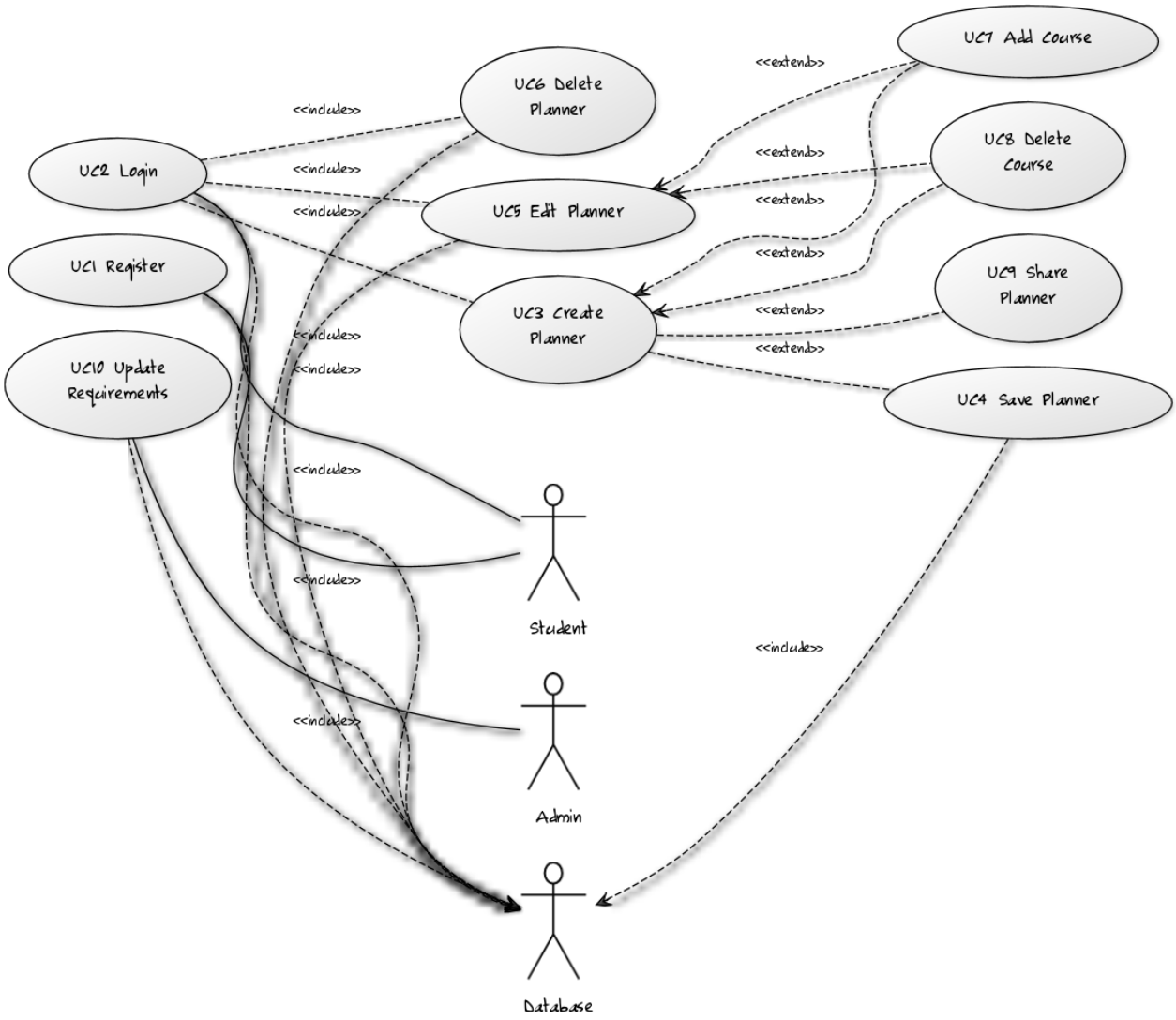
**Actors:**

- Student/User (Primary)
- Database (Secondary)
- Admin (Secondary)

**Use Cases:**

- Register
- Login
- Create Planner
- Save Planner
- Edit Planner
- Delete Planner
- Add Course
- Delete Course
- Share
- Update Requirements

## Use Case Diagram:



## Use Case Specifications:

| |
|---|
| **Header**<br><br>**Name: Register**<br>**Number: UC1** |
| **Description:** Allows student to register for the RUplanner website<br>**Purpose:** For student to register and create an account on the RUplanner website for saving and editing purposes<br>**Related Requirements**: REQ2<br>**Related Use Cases:** UC2<br>**Primary Actor**: Student<br>**Secondary Actor:** Database<br>**Technical Requirements:** Functioning website with database to insert newly registered student |
| **Body**<br><br>**Preconditions**: The student has not registered before and is registering with a new username/email and password<br>**Postconditions**: The student's login information is saved into the database for future retrieval<br><br>**Flow of Events for Main Success Scenario:**<br>-> 1. The student requests to register for the website<br><- 2. The system accepts registration due to unregistered username/email and password<br>**Flow of Events for Extension (Alternate Scenario):**<br>-> 1. The student requests to register for the website not using a unique username/email<br><- 2. The system shows that the username/email has already been taken, and requests for a unique registration username/password. |
| **Trailer**<br><br>**Issues:** Students must provide a unique username/email in order to gain access.<br>**Assumptions:** A student who has already registered will not try to register again using an identical username/email<br>**Design Comments:** Easy to integrate using databases<br>**Change Log:** N/A |

**Header**

**Name: Login**
**Number: UC2**

**Description:** Allows the user to login to his previously registered account
**Purpose:** To provide access to the websites functionality and start/reopen their planner
**Related Requirements**: REQ1, REQ2, REQ3
**Related Use Cases:** UC1
**Primary Actor**: Student
**Secondary Actor:** Database
**Technical Requirements:** Functioning website with database to check student can login

**Body**

**Preconditions**: The student has already registered for the website and is attempting to login.
**Postconditions**: The student has successfully logged in and can view their planner or start a new one

**Flow of Events for Main Success Scenario:**
-> 1. The student enters in their log in information
<- 2. The system accepts the information and displays their planner
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The student enters in wrong log in information
<- 2. The system shows log in information is incorrect and asks student to try again

**Trailer**

**Issues:** Keeping track of the login information for all users, verifying user correctly
**Assumptions:** User will eventually enter their information in correctly
**Design Comments:** Simple enough to do with databases
**Change Log:** N/A

**Header**

**Name: Create Planner**
**Number: UC3**

**Description:** Allows the student to create a planner
**Purpose:** Gives the student the opportunity to create their planner so they can see what courses need to be taken and other important information regarding graduation
**Related Requirements**: REQ1, REQ2, REQ3, REQ4
**Related Use Cases:** UC4, UC5, UC6, UC7
**Primary Actor**: Student
**Secondary Actor:** Database
**Technical Requirements:** Functioning website UI

**Body**

**Preconditions**: The student has successfully logged in and wants to start a new planner
**Postconditions**: The Student creates a blank planner after specifying major with which they can add courses too and edit/save for future use

**Flow of Events for Main Success Scenario:**
-> 1. The student specifies major, requests to create a new planner
<- 2. The system provides a new planner for the user
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The student requests for a new planner
<- 2. The system doesn't provide one, most likely a server or browser error

**Trailer**

**Issues:** Ensuring that a blank planner comes up with all available functions
**Assumptions:** No server error or browser error will hamper the action from taking place
**Design Comments:** Blank slate with functioning UI to add/drop classes
**Change Log:** N/A

**Header**

**Name: Save Planner**
**Number: UC4**

**Description:** To save an existing planner
**Purpose:** To allow students to save their current planner in order to revisit it at a later time
**Related Requirements**: REQ2, REQ3
**Related Use Cases:** UC3, UC5
**Primary Actor**: Student
**Secondary Actor:** Database
**Technical Requirements:** Saving the already created planner in a textual database for future use

**Body**

**Preconditions**: The Student must have logged in and already created a planner
**Postconditions**: The student's planner will be saved and can be revisited at a later time.

**Flow of Events for Main Success Scenario:**
-> 1. The student hits the save button on his planner
<- 2. The system indicates the planner was saved successfully
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The student hits the save button on his planner
<- 2. The system indicates the planner was not saved successfully, suggesting that there was a database or server error.

**Trailer**

**Issues:** Database/server/browser error not saving the planner correctly
**Assumptions:** User meant and correctly hit the save button on his browser
**Design Comments:** Saved  and stored in databases
**Change Log:** N/A

**Header**

**Name: Edit Planner**
**Number: UC5**

**Description:** Edits a previously saved planner
**Purpose:** To allow the user to go back to a saved planner to make changes
**Related Requirements**: REQ2, REQ3
**Related Use Cases:** UC3, UC4, UC7, UC8
**Primary Actor**: Student
**Secondary Actor:** Database
**Technical Requirements:** Editing/updating entries in the database

**Body**

**Preconditions**: The student wants to revisit a previously saved planner.
**Postconditions**: The student has reentered a previously saved planner for editing

**Flow of Events for Main Success Scenario:**
-> 1. The student logs in and requests to edit a saved planner
<- 2. The system reopens the saved planner for additional editing
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The student logs in and requests to edit a saved planner
<- 2. The system shows no previously saved planner or fails to open the planner, indicating incorrect saving of past planner or database error not retrieving the planner

**Trailer**

**Issues:** Finding the correct planner to reopen to edit
**Assumptions:** The user saved his previously planner correctly
**Design Comments:** Requires working database
**Change Log:** N/A

**Header**

**Name: Delete Planner**
**Number: UC6**

**Description:** Deletes a planner
**Purpose:** Allows the user to delete previously made planners or currently worked on planner
**Related Requirements**: REQ3
**Related Use Cases:** UC3, UC4, UC5
**Primary Actor**: Student
**Secondary Actor:** Database
**Technical Requirements:** Deleting an entry out of the database

**Body**

**Preconditions**: The user is currently working on a planner they no longer want or wants to delete a previously saved planner
**Postconditions**: The student's planner was deleted

**Flow of Events for Main Success Scenario:**
-> 1. The student clicks the delete button next to the planner they want to delete
<- 2. The system indicates said planner is deleted
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The student clicks the delete button next to the planner they want to delete
<- 2. The system doesn't delete the planner or deletes the wrong one.

**Trailer**

**Issues:** Potential issues with deleting the wrong planner or the planner not being deleted
**Assumptions:** Student selects the correct planner to delete
**Design Comments:** Database entry deletion
**Change Log:** N/A

**Header**

**Name: Add Course**
**Number: UC7**

**Description:** To add a course to a student's planner
**Purpose:** To allow students to fill their planner with future courses
**Related Requirements**: REQ3, REQ4, REQ5, REQ7, REQ8
**Related Use Cases:** UC6, UC8
**Primary Actor**: Student
**Secondary Actor:** Database
**Technical Requirements:** Functioning GUI with database of available courses

**Body**

**Preconditions**: The student has created a planner and wants to populate it with courses
**Postconditions**: The student has courses added to their planner.

**Flow of Events for Main Success Scenario:**
-> 1. The student selects the add course button and selects the correct course
<- 2. The system adds the course to the planner
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The selects the add course button and selects the incorrect course
<- 2. The system adds the incorrect course to their planner or indicates an error, such as duplicate courses or pre requisite issues.

**Trailer**

**Issues:** Graphical issues with adding courses or incorrect adding of credits/finding pre requisites
**Assumptions:** The user knows which course she wants to add
**Design Comments:** Full list of courses via database and dynamic requirement/credit GUI
**Change Log:** N/A

**Header**

**Name: Delete Course**
**Number: UC8**

**Description:** To delete a course in a student's planner
**Purpose:** To allow students to delete added courses in their planner
**Related Requirements**: REQ3, REQ4, REQ5, REQ7, REQ8
**Related Use Cases:** UC6, UC8
**Primary Actor**: Student
**Secondary Actor:** Database
**Technical Requirements:** Functioning GUI with courses on the planner

**Body**

**Preconditions**: The student has created a planner with courses and wants to delete a course.
**Postconditions**: The student's planner no longer shows the course.

**Flow of Events for Main Success Scenario:**
-> 1. The student selects the delete course button and selects the correct course
<- 2. The system deletes the course on the planner
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The selects the delete course button and selects the incorrect course or has no available courses to delete
<- 2. The system deletes the incorrect course on their planner or indicates an error that they have no available courses to delete

**Trailer**

**Issues:** Graphical issues with deleting courses or incorrect adjustment of credits/perquisite classes after course deleted
**Assumptions:** The user knows which course she wants to delete
**Design Comments:** Accurate planner GUI and dynamically adjusting credit/pre requisite information
**Change Log:** N/A

**Header**

**Name: Share Planner**
**Number: UC9**

**Description:** Shares the student's planner with others
**Purpose:** To allow the student the option to share their schedule via email or social media
**Related Requirements**: REQ6
**Related Use Cases:** N/A
**Primary Actor**: Student
**Secondary Actor:** N/A
**Technical Requirements:** Saving and sending the planner as an image

**Body**

**Preconditions**: The student has a planner with which she wants to share
**Postconditions**: The student successfully exported the planner as an image to another email or social media outlet

**Flow of Events for Main Success Scenario:**
-> 1. The student clicks the share button
<- 2. The system asks which method (facebook/twitter/email)
-> 3. The student selects the appropriate option
<- 4. The system exports the planner to an image file and correctly sends it
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The student clicks the share button
<- 2. The system asks which method (facebook/twitter/email)
-> 3. The student selects the appropriate option but uses incorrect data (e.g wrong email address or invalid facebook/twitter account)
<- 4. The system exports the planner to an image file and sends it to the incorrect destination

**Trailer**

**Issues:** Invalid data for email or facebook information
**Assumptions:** Student wants to share their data and correctly enters the right email/social media outlet account
**Design Comments:** Integration with email and facebook/twitter
**Change Log:** N/A

**Header**

**Name: Update Requirements**
**Number: UC10**

**Description:** Allows the admin to update the requirements for the engineering majors
**Purpose:** Due to university changes in major requirements, periodic updating of the requirements for each engineering major is required in order to not fall out of date
**Related Requirements**: REQ9
**Related Use Cases:** N/A
**Primary Actor**: Admin
**Secondary Actor:** Database
**Technical Requirements:** Database access to modify requirements

**Body**

**Preconditions**: The requirements for a major has changed and the database needs to reflect those changes
**Postconditions**: The database is successfully changed to reflect new requirements

**Flow of Events for Main Success Scenario:**
-> 1. The Admin inserts/modifies data into the database
<- 2. The database accepts the changes and reflects the changes
**Flow of Events for Extension (Alternate Scenario):**
-> 1. The Admin inserts/modifies data into the database
<- 2. The database rejects the changes, probably because the user didn’t correctly type in the admin password or didn’t have access

**Trailer**

**Issues:** N/A
**Assumptions:** Person who is trying to make changes is an admin with administrative access
**Design Comments:** Updating databases
**Change Log:** N/A

## Data Dictionary (alphabetical order):

**Admin**: A user who maintains the database and has access over the website, including updating database information and accounts.

**Database**: Where all information about a user's course planner will be stored and called upon for future use/deletion.

**Course List**: A list of all available courses which users can quickly search through

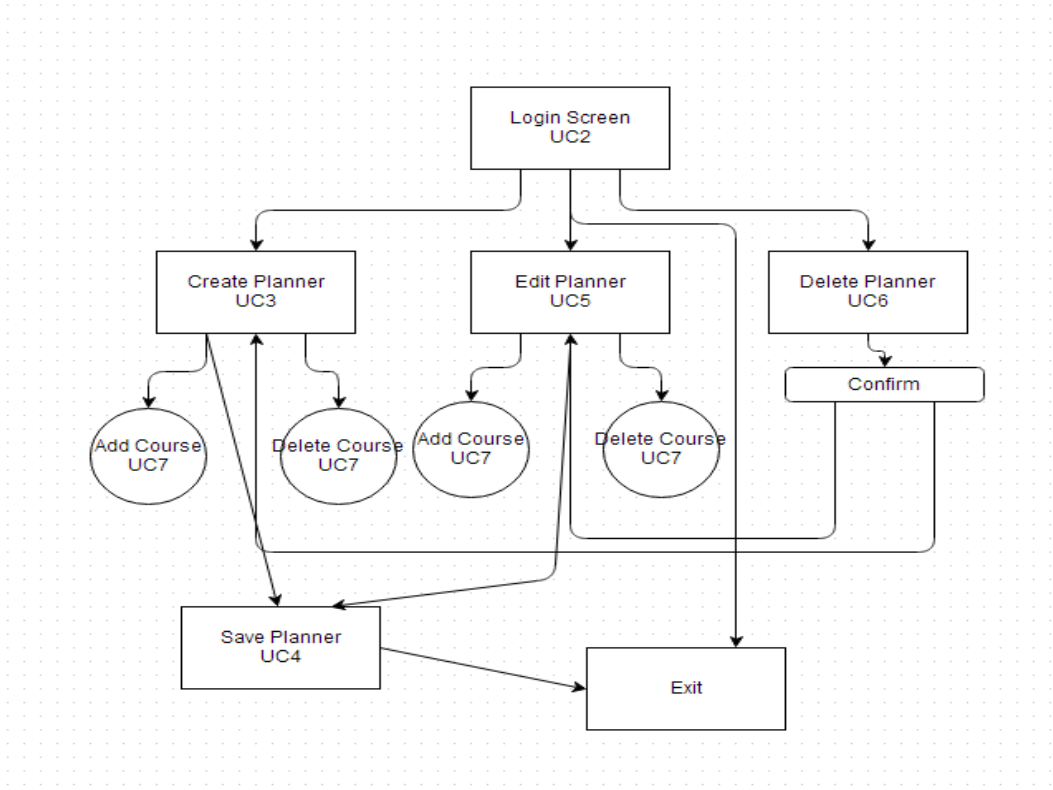
**GUI**: The graphical user interface for the planner with which student's will be able to create/edit/delete their planner

**Planner**: A user's online list of courses to take.

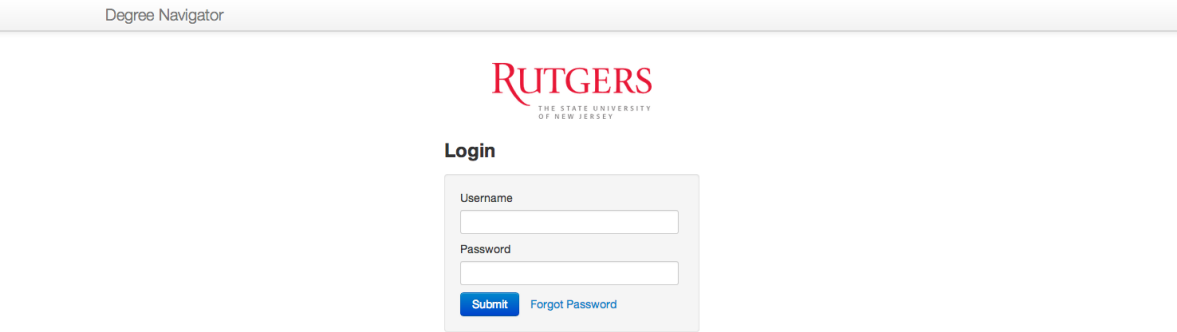
**User/Student**: Person using the website for planning purposes

## User Interface Specification:

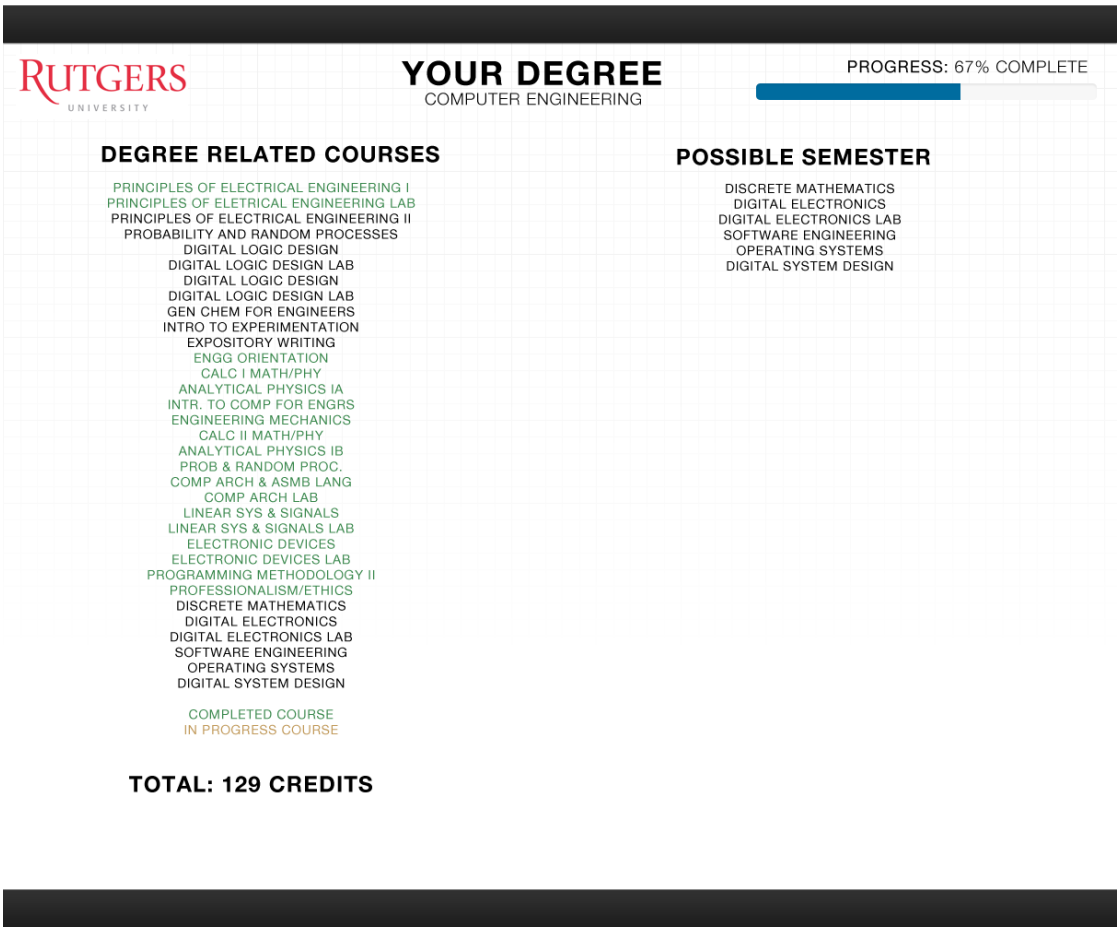
Included is the User Interface chart and 2 sample images of the (early staged) login screen and requirements page pertaining to the planner.



## Login Sample Screenshot:




## Sample Requirements page for planner:



**Deliverable Chart:**

| UC# | Deliverable I (3/30/2013) | Deliverable II (3/30/2013) | Deliverable III (3/30/2013) |
|---|---|---|---|
| UC1 | ✓ | ✓ | ✓ |
| UC2 | ✓ | ✓ | ✓ |
| UC3 | × | ✓ | ✓ |
| UC4 | × | ✓ | ✓ |
| UC5 | × | ✓ | ✓ |
| UC6 | × | ✓ | ✓ |
| UC7 | × | × | ✓ |
| UC8 | × | × | ✓ |
| UC9 | × | × | ✓ |
| UC10 | ✓ | ✓ | ✓ |

## Rationalize Schedule

**Resources**
The team is using github repo to allow easy updates of code. Steve is using xcode IDE for html/css/java script programming while Nick is using visual studio and mysql for database and back end programming.

**Expertise**
All team members have experience programming, however Steve is much more familiar with interfaces and front end design, while Nick has experience with backend databse programming. GUIs in HTML/CSS/Javascript will be programmed by Steve, while backend programming and databases will be maintained by Nick.

**Risks**
While there are no dire risks, getting social media integration may prove troublesome, and ensuring a fluid and elegant GUI is paramount to the task at hand. This will be as much of an aesthetic design challenge as it will be programming and data management challenge.

## Team Responsibilities:

**Steve**: In charge of website design, including HTML/CSS/Javascript work and making the site fluid and usable.

**Nick**: In charge of back end programming (algorithms, computations) and database design/maintenance to tie in with the Steve's front end design.