Visual Course Planner

Manual and User Documentation

Version 1.0
22.04.2019

Table Of Contents

1 Quick Start

For Devs:

1. Clone the repository: https://github.com/UBCO-COSC499-Winter-2018-Term-1-2/visual-course-planner
2. Ensure a MySQL instance is running on port 3306.
3. Create a database called vcp, and a database user:
    a. Username: vcpUser
    b. Password: (no password)
    c. Host: localhost
    d. Grant the user all privileges to the vcp table.
4. Run the `database.sql` script on the vcp database to create all necessary tables.
5. Install npm and NodeJS version 10 https://nodejs.org/en/. (packaged together)
6. Navigate to the project root directory in a terminal and run: `npm install && (cd client && npm install)`.
7. `npm run dev` to start the app.
8. localhost:3000 should open in your browser.

For Users:

1. Navigate to the signup page of the vcp (/signup).
2. Fill in the required details.
3. Click "Create Account".
4. Go to the email you provided and click the link to verify your email (check you junk folder).
5. You should be taken to the main page of the VCP.
6. Click create a plan to create a new plan.
7. Pick the degree and specialization you want for this plan and hit submit.
8. Add a few terms by clicking add term in the top right.
9. You can now add courses by opening the add course menu (click add course).
10. Click and drag a course into a term.
11. Your plan will automatically be saved.

2 INTRODUCTION

2.1 INTRODUCTION

Organizing a degree plan at the University of British Columbia is a tedious and confusing task with the current way that degree planning is organized. A static page for each program features a list of required courses to be taken to fully complete the program. A student may browse all courses on the Course Description page, however not all courses are guaranteed to be offered every year, or even at all.

The Visual Course Planner (VCP) is designed to assist students or academic advisors to plan an entire degree at the University of British Columbia Okanagan in one application.

The VCP is visually pleasing featuring a contemporary UI design and drag and drop controls. The VCP enables students to create an up-to-date degree plan, removing any stress of missing credits or empty credits (credits that do not count for their degree).

2.2 PURPOSE OF DOCUMENT

This document outlines the underlying structure and how to get the best usage of the Visual Course Planner.

2.3 MISSION STATEMENT

The most efficient way for planning your future at UBC Okanagan.

2.4 SYSTEM OVERVIEW

The Visual Course Planner (VCP) is a web application that allows students and academic advisors to plan degrees at the University of British Columbia Okanagan campus. The design of the Visual Course Planner is to enhance the students ability to successfully plan their degree at UBCO without having to worry about missing credits and taking courses that don't apply to their degree.

The VCP features a total of four user interfaces: Login and Signup, Student Information, Admin Portal, and the Planner Interface. The VCP supports all major web browsers.

There are only two types of users (admins or students) who can create an account to interact with the VCP. Admins and Students login using the same login page.

Admins have the ability to upload a file containing the current offered courses and program requirements based on what the University is offering. Admins have the ability to upload more than one file at a time. Once the data has been uploaded it will then be displayed and accessible by each student when they are making a degree plan. Admins will not have any access to user's accounts. The information available to the user depends on what the admins upload. Unless an admin indicates a course is offered, users will not be able to add that course to their degree plan.

Students will be able to create an account by providing their personal information such as their first name, last name, email and their chosen password.

After, an confirmation email will be sent to the email address that was provided. Once a student account is created and confirmed via the confirmation email, they will then be prompted to fill in additional information about themselves such as which degree they wish to enroll in, desired major, and previously completed courses. A degree *must* be selected in order for the student to proceed to the next step. Whilst the student is

deciding on a degree, a description of what the program entails is presented on the right hand side of the screen. The description presented is provided by UBC.

Once a student has provided their information, they will be taken to the Visual Course Planner interface. On the left side of the planner interface is a static vertical menu that displays the student's name, a logout button, a save plan button, a list of previously saved degree plans, a "favourite plan" button, a "new plan" button, and a note section. On the far right is a minizable menu that displays a search bar, present warnings based on a current change, and a description of a previously selected course. Across the top of the window is an optimize button, and a warning summary. The optimize button will shrink the students entire degree plan to the shortest duration possible based on which courses are included.

The main component of the VCP is located in the middle and features the courses which the student plans to take during their time at UBCO. Courses are organized by year and then by semesters. If summer courses, Go-Global and or Co-op courses are added to the degree plan, corresponding categories will be added (i.e. Summer Courses; Term 1, Term 2). Here, they can utilize drag and drop functionality to drag a courses into their plan depending on the term the course is offered. All required courses are outlined in a salmon pink colouring while the elective courses are outlined in electric blue. Due to the limitation of screen space, the area that displays all courses for a degree plan is scrollable rather than scalable to avoid shrinking all elements into the restricted area.

This web application will follow the MVC structure and use a REST API. The UI will make request JSON data through a rest call to the server. The JSON data depends on the type of event triggered. The server queries the database and builds a JSON object from the query results. Once the UI receives the JSON, it will update the view accordingly.

A data scraper will retrieve course data from the UBC Okanagan academic calendar. This data will be formatted and imported into the system.

For a detailed description on how the VCP is organized please see section 3: Organization

3 SETTING UP THE ENVIRONMENT

The following section describes the necessary steps taken to properly setup your environment for the Visual Course Planner.

Development on the backend of the Visual Course Planner is primarily coded with Node.js; the front end utilizes React and CSS. A basic understanding for each language is recommended in order to alter any features implemented in the VCP.

3.1 INSTALLING NECESSARY PROGRAMS AND PACKAGES

To get started, we recommend installing Homebrew on macOS (or use apt-get on Linux) for an easy set up: https://brew.sh. If you do not want to install Homebrew you will have to manually install all programs mentioned to update the Visual Course Planner.

Once Homebrew is installed, you can type `brew install npm` in terminal window to install npm. Alternatively to download npm and node, you. can visit https://nodejs.org/en/ which will download npm and node together.

2.2 CONNECTING TO THE VCP DATABASE

In order to proceed to the next step, make sure the MySQL 8.0.15 or higher is installed
https://dev.mysql.com/downloads/mysql/

Run a MySQL instance on port 3306. You can use a tool like phpMyAdmin to do these next steps, or run commands directly on a terminal after typing mysql.

1. A user will need to be created with the name vcpUser and no password on localhost.
2. Create a database called vcp. (`CREATE DATABASE vcp; USE vcp;`)
3. Grant that user all privileges on the vcp database (`GRANT ALL PRIVILEGES ON vcp.* TO 'vcpUser'@'localhost';`)
4. Run the `database.sql` script (located in the root directory) on the vcp database to create all necessary tables.

3.1.1 GITHUB REPOSITORY

To connect to the VCP Git Repository you will need to be added as a contributor. You can find the repository at:
https://github.com/UBCO-COSC499-Winter-2018-Term-1-2/visual-course-planner

The root folder contains all code and configuration files for the Visual Course Planner.

1. Clone the repository.
2. Navigate to the project root directory in a terminal and run: `npm install && (cd client && npm install)`.
3. Run `npm run dev` to start the app.
4. localhost:3000 should open in your browser.

3.1.2 DEPENDENCY REQUIREMENTS

Dependencies are automatically installed via npm in the previous steps.

4 VCP ORGANIZATION

The git repository has 3 main areas. The root folder, the client folder, and the server folder.

The root folder contains configuration files such as TravisCI build config, database user information, linting rules, and the database script.

The client folder contains all frontend development work, while the server folder contains all backend nodejs/expressjs code. The api is organized by matching the folder hierarchy to the route path (e.g. the GET /api/users route is located in the server/src/routes/api/users.js file).

The front end development of the Visual Course Planner is organized in two categories: Components and Containers. The Component section contains different frontend features implemented on the VCP website. The Container section holds elements for Index and the Main interface.

4.1 CLIENT COMPONENTS

The following are components used for the frontend development of the Visual Course Planner, location in client/src/components.

Component	Files	Description
Admin Portal	AdminPortal.css	The style sheet for all elements/React components featured in the Admin Portal Interface
	AdminPortal.js	All elements/React components featured in the Admin Portal Interface
Backdrop	Backdrop.css	The style sheet for the Backdrop component
	Backdrop.js	A React Component that dims the background once the Course List Sidebar slides in & is visible on the planner page . Upon clicking the backdrop, the Course List Sidebar collapses and the focus is on the planner page.
Backdrop Button	BackdropButton.css	The style sheet for the BackdropButton component.
	BackdropButton.js	React Components for a backdrop button. Toggles a
Button	button.css	The style sheet for the general button used throughout the Visual Course Planner
	button.js	A general button used throughout the Visual Course Planner
Confirm Email	ConfirmEmail.css	The style sheet for the ConfirmEmail component
	ConfirmEmail.js	React component for the Confirm Email page
CourseInfoDisplay	CourseInfoDisplay.css	The style sheet for CourseInfoDisplay
	CourseInfoDisplay.js	React component that display the Course's Titles and Courses Descriptions
CourseSearchBar	CourseListSideBar.css	The style sheet for the CourseSearchBar
	CourseListSideBar.js	React Component for the Search Bar used to search up courses currently offered at UBCO. This component is rendered inside the CourseListSideBar component.
Course	Course.css	The stylesheet that controls how a course item is displayed inside the Term component.
	Course.js	React component for course items (e.g COSC 111)
CourseListSideBar	CourseListSideBar.css	The style sheet that controls how the list of courses are displayed in the Course List Sidebar
	CourseListSideBar.js	React Component that displays a list of all courses available on the Visual Course Planner. This component is a side bar and slides in from right to left on the planner page. It is activated by clicking the “Add Course” button
Favicon	vcp-favicon[final.jpg	The favicon for the Visual Course Planner website

FavouriteBtn	FavouriteBtn.css	Stylesheet for the FavouriteBtn component. Responsible for turning the “Heart” button red or gray based on its state
	FavouriteBtn.js	React Component for the Heart icon featured on the Planner page
FilterMultiSelect	multiSelectMenu.css	The style sheet for the Multi Filter Select
	multiSelectMenu.js	React Component to create a multi filter selection for which courses the student has already taken.
Input	Input.css	The stylesheet for all types of input used throughout the VCP.
	input.js	Checks to see what type of input field is called and displays specific context them accordingly. Also checks if the element has been touched by user and displays a red border if input text field is left empty.
Login	LoginInterface.css	The style sheet for all elements for the Login Menu
	LoginInterface.js	React components for the Login Menu
NewPlan	NewPlan.css	The style sheet for the Degree Plan component
	NewPlan.js	React component that creates new Degree Plan for a student account
Notes	NoteArea.js	React Component for the Personal Note area designated for each Degree Plan the student makes. It is rendered in the left side bar
Optimize Button	OptimizeBtn.css	The style sheet for the Optimize Button
	OptimizeBtn.js	React component for the Optimize button. Functionality not implemented.
Plan List	PlanList.css	Style sheet for the PlanList component
	PlanList.js	React component for the list of plans a user has. This is rendered in the left sidebar on the planner page.
PlanName	PlanName.css	Style sheet for the PlanName component
	PlanName.js	React Component for the name of the plan. It is rendered on the top bar (planner header) on the planner page.
Planner	NewPlanButton.js	React Component for the create new plan button that creates a new Degree Plan
	PlannerArea.css	The style sheet for PlannerArea component
	PlannerArea.js	React Component for where the user interacts with their plan. Planner Area contains other React components. This is the main component where the user can add terms, add courses and use the drag & drop feature.

Planner Header	PlannerHeader.css	The style sheet for PlannerHeader component
	PlannerHeader.js	React Component functional component with no internal state (wrapper for any items in the header)
Previous Courses	PreviousCourses.css	The style sheet for the PreviousCourses component
	PreviousCourses.js	React component for the page that allows users to enter their course history and the record of which courses a student has previously taken and accepted by UBCO
Session	Session.css	Style sheet for the Session component
	Session.js	React component that creates and renders an academic session on the planner page. For example: 2018W or 2019S. Each session contains terms (Term components) within itself.
Signup	SignupInterface.js	React Component for the sign up page for a new user.
StudentInfo	StudentInfo.css	The style sheet for StudentInfo component
	StudentInfo.js	React Component that groups and displays the Student's information such as their name and current standing. The Edit Personal Info and Log Out button are also included in this React component
Term	Term,css	The style sheet for the Term component
	Term.js	React Component that create a Term container with the Term ID, and Term heading. This component is rendered within the Session component. This is the component where users can drag and drop a Course item.
UserProfile	profile.css	The style sheet for UserProfile component
	profile.js	React component that renders a User's Profile page.
Warning Snackbar	WarningSnackbar.css	Style sheet for WarningSnackbar component
	WarningSnackbar.js	React component that renders a popup snackbar on the page, close to the bottom of it. This is the container that shows warnings regarding the plan and is only visible when the "Warnings" button is clicked.
Warning Summary	WarningSummary.css	Stylesheet for WarningSummary component
	WarningSummary.js	React component for the warning summaries. This component is rendered as the "Warning summary" button in the Planner Header component on the planner page. Clicking this button displays the WarningSnackbar component.

4.2 CLIENT CONTAINERS

The following are containers used for the frontend development of the Visual Course Planner located in client/src/containers.

Container	**Files**	**Description**
Main	Main.css	The style sheet for all elements featured on the Main Interface. Includes the CSS grid layout for the main page.
	Main.js	Parent React component that contains all the other components, such as: degree plan, degree notes, degree name, favourite and optimize button e.t.c
Sidebar	Sidebar.js	A stateless react component to add a sidebar element
Sidebar Area	SidebarArea.js	A stateless react component to divide up the sidebar into groupings of components

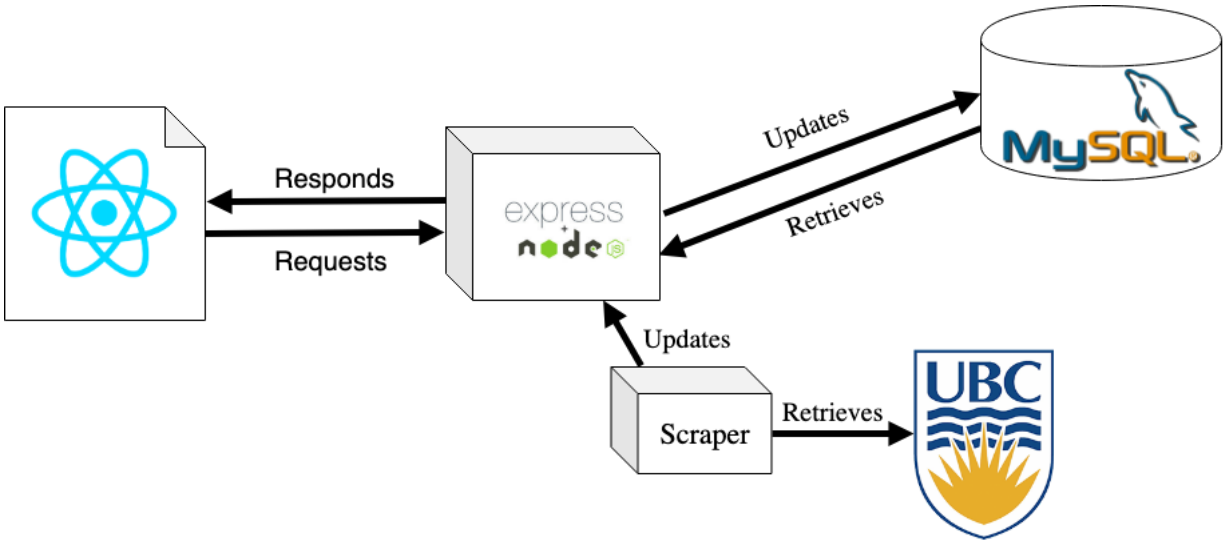
4.3 SERVER COMPONENTS

Component	**Files**	**Description**
Routes	users.js terms.js warnings.js specializations.js sessions.js plans.js degrees.js courses.js admin/upload.js	Located in server/src/routes/api. API routes that handle any HTTP requests to /api. Provides a way for the client (React) to make requests to the server, and access information in an asynchronous fashion.
Models	User.js Term.js Course.js Degree.js Session.js Plan.js Specialization.js	Files that handle database calls. Provides a away of abstracting the database calls and exposing a simple api. To make a database call, import the required model and call the necessary function from the model. Called by the code that handles the routes if they need to access the database.
Services	CourseService.js WarningService.js SessionService.js SpecializationRequirementsService.js TermService.js EmailService.js	Files that contain logic for server-side processing of information. Enables the logic to be abstract from any http or database request, and thus encourages ease of unit-testing.
Course Scraper	coursescraper.js	A scraper that retrieves all course data from the UBC academic calendar.
Server	server.js	Starts an express server that handles all

		incoming HTTP requests to localhost:5000.

4.4 ARCHITECTURE

4.4.1 Architecture Diagram



As you can see in the diagram, the architecture is fairly straightforward. The view (React) makes requests to the API, and is updated based on the information it receives. The API updates the database. The scraper gets information from the official UBC academic course calendar sends the information to the backend, which in turn updates the database.

4.5 ADMIN PORTAL

The admin portal is located at localhost:300/main on a local instance. You must be logged in as an administrator to view the page. Currently to make other users administrators the database must be modified directly. Here the admin may upload either currently offered courses or specialization requirements.

4.5.1 Offered Courses

Select this option to upload the currently offered courses. The CSV should be formatted like so:
The first line of the file must be this header:
`COURSE_CODE,TERM,ACADEMIC_YEAR`

COURSE_CODE is the course identifier e.g. COSC 111.
TERM is the term number the course is offered in. Must be 1 or 2.
ACADEMIC_YEAR is the session the course is offered in. Must in the format YEAR + SEASON. E.g 2018W. Year is 4 digits long, season is a single letter: either W or S (winter or summer).

Example file:
`COURSE_CODE, TERM, ACADEMIC_YEAR`

```
COSC 111, 1, 2018W
COSC 121, 2, 2018W
COSC 304, 1, 2019W
COSC 222, 1, 2018W
COSC 310, 2, 2018W
COSC 211, 1, 2019W
MATH 221, 1, 2018W
MATH 100, 1, 2019W
MATH 222, 2, 2019W
```

4.5.2 Specialization Requirements

Specialization requirements are the requirements of a chosen major or minor, such as a major in Computer Science. The file must be in CSV format with the following header:
`CREDITS,COURSES,EXCEPTIONS`
Credits is how many UBC credits are needed from the following courses (each course is assumed to be 3 credits)
Courses is which courses may be applied to a particular specialization. If there are multiple courses associated with a particular requirement, the list of course must be enclosed in quotes.
Courses may also be a category in the format [UPPER] (COURSE CODE | AREA | GENERAL). A space separated string of either 1 or 2 words. The first word, if it exists must be UPPER. The second word may be a course code (e.g. COSC) or area (e.g. ARTS or SCIENCE) or GENERAL (anything).
Exceptions is which courses may not be applied to a particular specialization.

Example file taken directly from the academic calendar):

```
CREDITS,COURSES,EXCEPTIONS
3, "ENGL 112, ENGL 114",
3, "ENGL 113, ENGL 150, ENGL 151, ENGL 153",
6, "MATH 100, MATH 101",
3, "PHYS 111, PHYS 112",
3, "PHYS 102, PHYS 121, PHYS 122",
3, "CHEM 111, CHEM 121",
3, "CHEM 113, CHEM 123",
3, "COSC 111, COSC 123",
3, COSC 121,
9, "COSC 211, COSC 221, COSC 222",
3, "MATH 200, MATH 221",
3, STAT 230,
3, COSC 304,
3, COSC 310,
3, COSC 320,
3, COSC 341,
3, PHIL 331,
15, UPPER COSC,
6, UPPER SCIENCE,
9, ARTS,
3, UPPER GENERAL,
21, GENERAL,
```

4.6 QUALITY PLAN

To ensure product quality a git branching workflow and code merging pattern is enforced.

4.6.1 Branching Strategy

During the development of the Visual Course Planner a workflow of feature branch → dev → master is used for maintained a high level of code quality and to enable code reviews. This improves features by enabling collaboration and helps to reduce knowledge silos.

3.4.1.1 Master

The master branch (“master”) contains production ready code, that has been fully tested according to the quality plan. This is where code is deployed to the remote server. Pull requests to master need to be approved by the Integration Lead before being merged. Only pull requests from dev will be merged into master, except in the case of hotfixes. PRs need to pass CI builds in TravisCI.

4.6.1.1 Dev

The dev branch (“dev”) is where features are merged together and tested in CI. Merges to dev will need to be approved by at least one other team member, and need to build in CI before being merged..

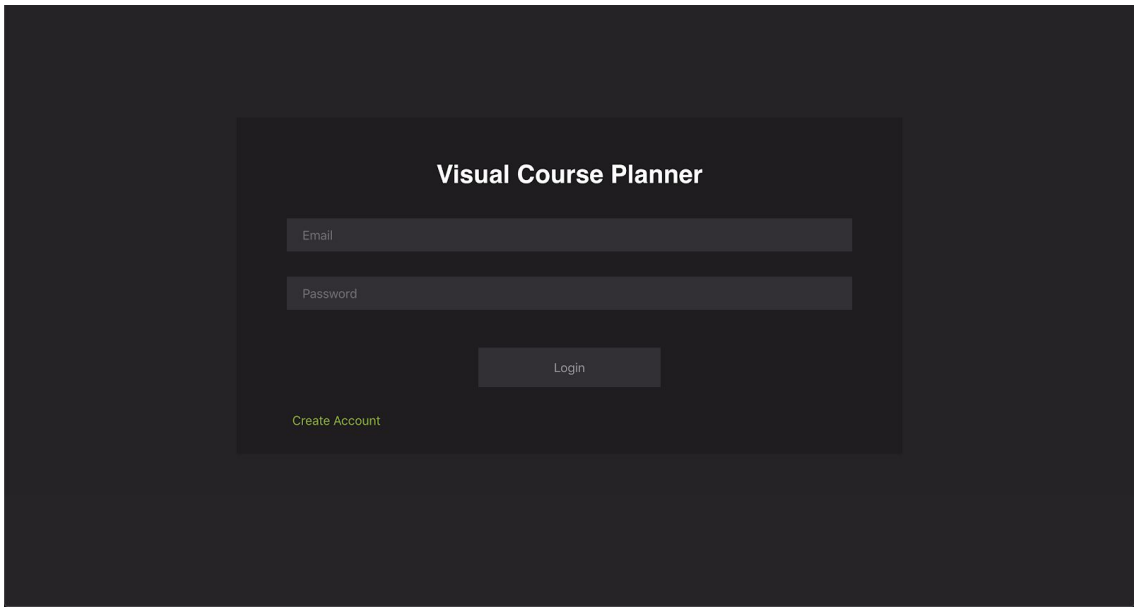
4.6.1.2 Feature branches

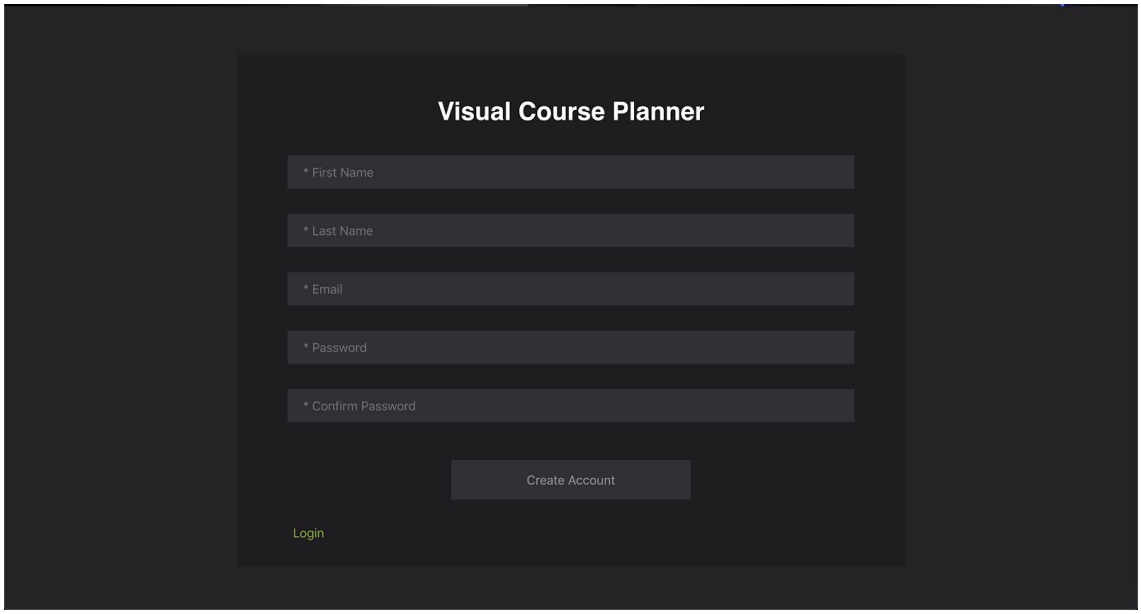
Feature branches are prefixed with “feature/”. Each branch contains an independent feature of the system. All pushes to a feature branch automatically trigger a build of the branch in CI.

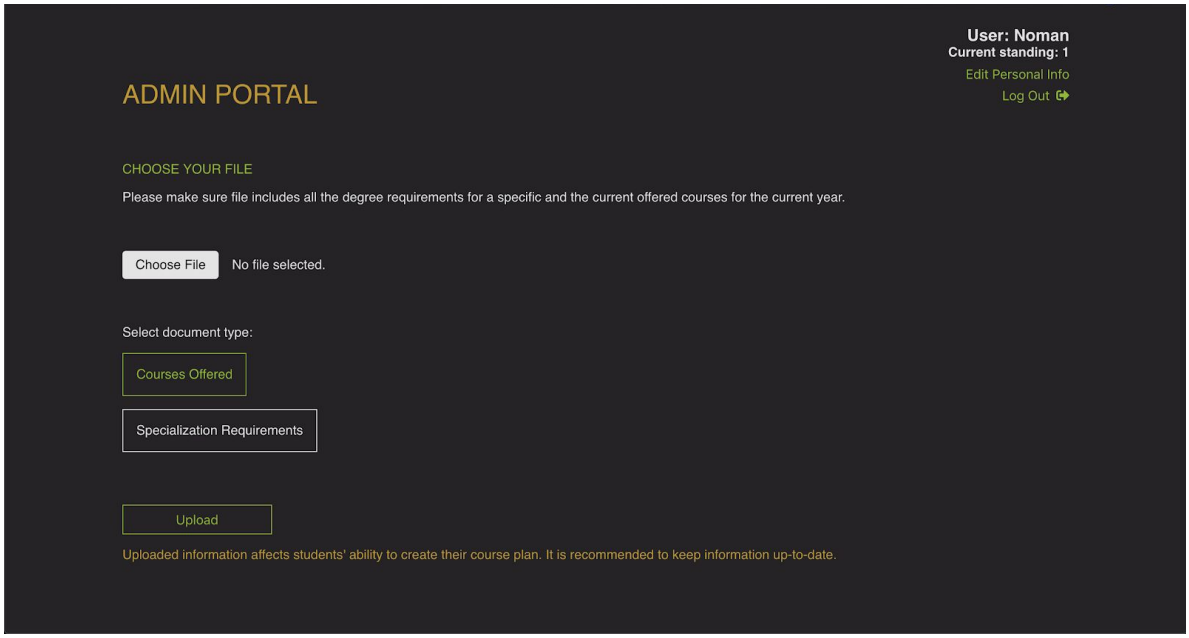
4.7 TECHNICAL SPECIFICATION

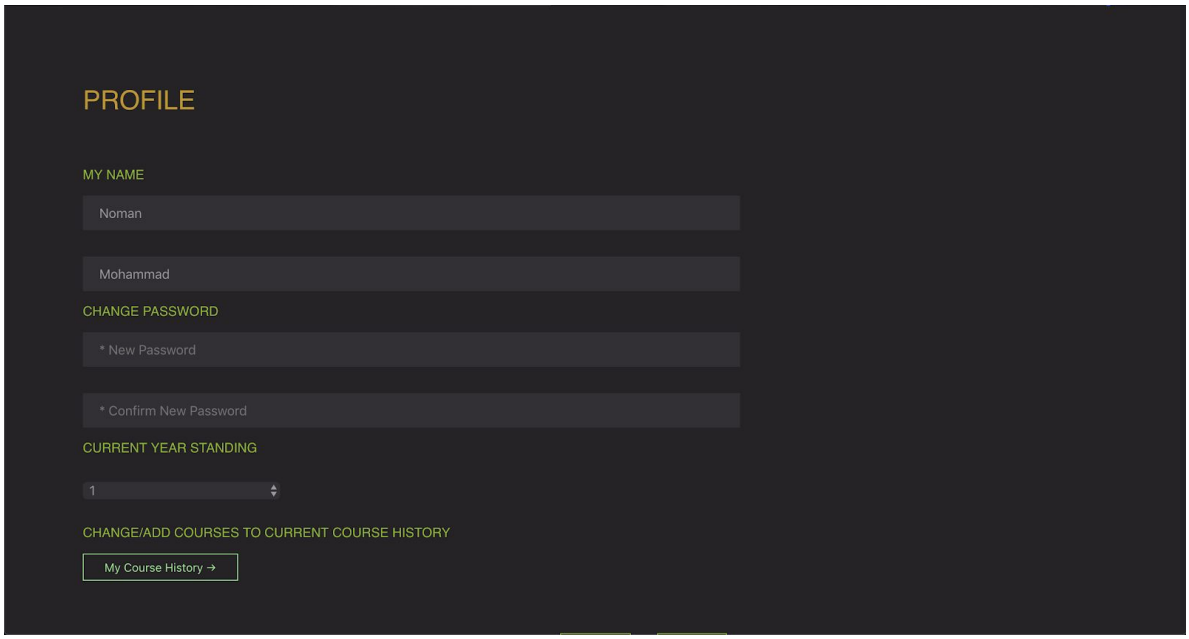
The VCP is built using MySQL, Express.js, React, and Node.js. The codebase is stored remotely on GitHub. Continuous Integration is provided by Travis CI.

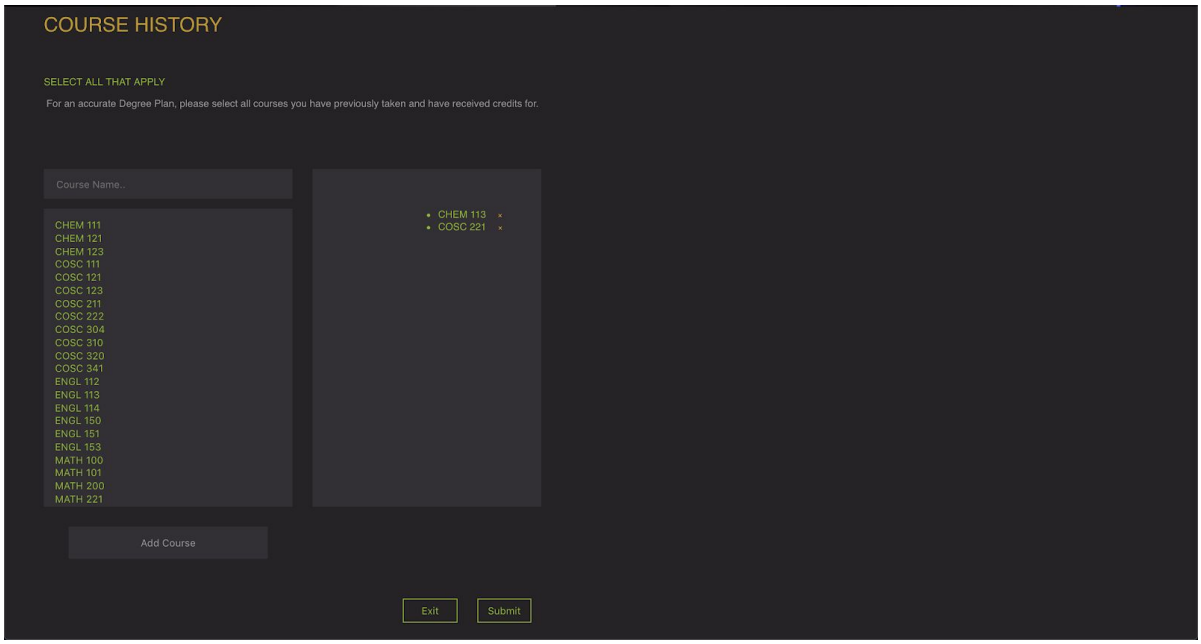
3.6 SITE IMAGES






User: Noman
Current standing: 1
Edit Personal Info
Log Out
ADMIN PORTAL
CHOOSE YOUR FILE
Please make sure file includes all the degree requirements for a specific and the current offered courses for the current year.
Choose File
No file selected.
Select document type:
Courses Offered
Specialization Requirements
Upload
Uploaded information affects students' ability to create their course plan. It is recommended to keep information up-to-date.


PROFILE
MY NAME
Noman
Mohammad
CHANGE PASSWORD
* New Password
* Confirm New Password
CURRENT YEAR STANDING
1
CHANGE/ADD COURSES TO CURRENT COURSE HISTORY
My Course History →


COURSE HISTORY
SELECT ALL THAT APPLY
For an accurate Degree Plan, please select all courses you have previously taken and have received credits for.
Course Name..
CHEM 111
CHEM 121
CHEM 123
COSC 111
COSC 121
COSC 123
COSC 211
COSC 222
COSC 304
COSC 310
COSC 320
COSC 341
ENGL 112
ENGL 113
ENGL 114
ENGL 150
ENGL 151
ENGL 153
MATH 100
MATH 101
MATH 200
MATH 221
CHEM 113
COSC 221
Add Course
Exit
Submit


User: Noman
Current standing: 1
Edit Personal Info
Log Out
Degree Plans
Favourites
Bachelor of Science
Plans
Bachelor of Arts
Create New Plan
Notes
My plan!
Bachelor of Science
Optimize
Warnings (0)
Add Course
2018W
2019S
2019W
Term 1
Term 2
Term 1
Term 2
Term 1
Close
Search for courses...
COSC 111
COSC 121
COSC 304
COSC 111
COSC 121
COSC 304

5 REMAINING TESTING

The following are features of the Visual Course Planner that have remaining testing that has not been done before delivery of April 17, 2019.

5.1 RESPONSIVE DESIGN

The Visual Course Planner is designed to work on major web browsers only. Viewing the VCP on a mobile device is possible but the Degree Plans cannot be edited or changed in any manner

6 WORKFLOW

The VCP team employed the Kanban methodology for issue tracking and completion. Team communications were done through Slack.

Team members created issue tickets on YouTrack and added them to the project board. Tickets had a system of hierarchy where a feature was created as a parent issue and its smaller components were created as smaller child issues.

The issue tickets were first added to the Backlog list and were picked by a team member to work on. Tickets would move to the Develop list. The ticket would be moved to the Review list for code review and once it was reviewed and merged into the Dev branch, the ticket would be moved to the Done list.

7 DEPLOYMENT ENVIRONMENTS

The VCP is deployed to a Digital Ocean droplet (2GB RAM) running Ubuntu 16.04. The server runs an nginx reverse proxy with a free Lets Encrypt SSL cert providing https protection. The NodeJS instance is managed by pm2 to provide support for restarting the app should it crash, as well as error log management.