CSMC Website Test Plan

Group Project

SE 6387

Submitted to

Prof. Neeraj Gupta

Associate Professor

Department of Computer Science,

The University of Texas at Dallas,

Richardson, TX -75080

Team Members:

Kiranmai Seemakurthi (kns180002)

Heta Shah (hxs180029)

Vatsal Batavia (vvb180000)

Mihir Hindocha (mxh170027)

Supriya Adhanki (nxa180045)

Harish Kaza (hxk180013)

Nikhita Rama Karkera (nxk180006)

Harry Tran (txt171930)

Chun-Ping Yang (cxy180010)

May 08, 2020

Table of Contents

1. Test Plan Identifier

- Identifier: CSMC:TestPlan:Spring2020

2. References

- Requirement Engineering Course Website: http://utdallas.edu/~chung/RE/syllabus.htm
- For Software Project Plan – http://www.utdallas.edu/~chung/SP/SoftwareProjectManagementPlanTemplate.htm
- Scrum Model image - https://en.wikipedia.org/wiki/Scrum_(software_development)#/media/File:Scrum_process.svg
- Software Architecture Document - https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm
- IEEE Test Plan Outline - https://jmpovedar.files.wordpress.com/2014/03/ieee-829.pdf

3. Introduction

This is the Test Plan document for the CSMC project. The primary focus of this plan is to ensure that the new CSMC system provides the same functionalities and improved functionalities as the current system. The project will have three levels of testing, Unit, System/Integration and Acceptance. The details for each level are addressed in the approach section and will be further defined in the level specific plans. The estimated timeline for this project is very aggressive (4 months (1 semester)), as such, any delays in the development process could have significant effects on the test plan. The acceptance testing is expected to take one week (last week) from the date of system delivery from system test and is to be done in parallel with the current application process.

4. Test Items

4.1 Unit Testing

Role	Module	Controller	Service	Repository
Student				
	Register	/student/session/schedule	getScheduleDetails()	findSectionAndCourseByStudentUsername(@Param("netid") String netid), findScheduledSessionBySectionIdAndStartTimeAfterToday(@Param("sectionid") UUID sectionid, @Param("today") Date today), getQuizDetails(@Param("sectionid") UUID sectionid, @Param("today") Date today)
		/student/session/schedule/sessions/{sessionId}	getScheduledSessionDetails(ModelMap model, UUID sessionId)	findById(ID var1)
		/student/session/schedule/unregister/{sessionId}sessionId}	removeRegisteredSession(String sessionID)	findById(ID var1), deleteById(ID var1)

		/student/session/register/timeslot/{tid}tid}	getSessionTimeslotDetails(UUID timeslotId)	findById(timeslotId)
		/student/session/register/timeslot/confirm/{tid}	confirmTimeslotRegistration(UUID timeslotId)	findById((ID var1), findByUsername(String username)
	Courses	/student/course/section	getCourseDetails()	getCourseDetails(@Param("netid") String netid)
	Grades	/student/session/history	getScheduledSessions()	getScheduledSessions(@Param("netid") String netid), getSessions(@Param("res") List<UUID> res)
			getQuizAttendance()	getQuizAttendance(@Param("netid") String netid)
			getWalkinAttendance()	getWalkInAttendance(@Param("netid") String netid)
Mentor	Staff Schedule	/mentor /schedule/weekly, /mentor/ schedule/weekly/{date}	scheduleWeeklyAction(ModelMap model, Date date)	findSemesterByActive(boolean active), findSessionTimeSlotsForEntireDayByDate(@Param("startOfDay") Date startOfDay, @Param("endOfDay") Date endOfDay), findQuizzesForEntireDayByDate(@Param("startOfDay") Date startOfDay, @Param("endOfDay") Date endOfDay), findByShowOnCalendar(boolean showOnCalendar)
	Calendar	/mentor/session/schedule	getSessionInformation()	getSessionInformation(@Param("today") Date today)
			getQuizInformation()	getQuizInformation(@Param("today") Date today)
		/mentor/session/view/{quizId}quizId}	getQuizDetails(UUID quizId)	getQuizDetails(@Param("quizid") UUID quizid)
		/mentor/session/timeslot/view/{timeSlotId}	getSessionDetails(UUID timeSlotId)	findById(ID var1)
		/mentor/swipe/session/{timeSlotId}timeSlotId}	getSwipeSessionDetails(UUID timeSlotId)	findById(ID var1)
		/mentor/swipe/ajax/session	getSessionTimeSlotDetails(UUID timeSlotId, String cardId)	findById(ID var1), findByCardId(S var1), findOnDuty(@Param("user") User user, @Param("today") Date today), save(S var1),

				save(S var1), findByUserAndTimeSlot(S var1, S var2)
	Swipe Screen	/mentor/swipe/ajax/walk_in	walkInSwipe(String scancode)	findByCardId(String scancode), findOnDuty(@Param("user") User user, @Param("today") Date today), save(S var1), findCurrentUser(@Param("user") User user, @Param("today") Date today),(@Param("user") User user, @Param("today") Date today), findCurrentUser(@Param("user") User user, @Param("today") Date today),(@Param("user") User user, @Param("today") Date today), findAll(), findCourses(),
		/mentor/swipe/ajax/entry	getSwipeEntryDetails(String topic, String courseId, String activityId, String quizId, String userId	findById(ID var1), findById(ID var1), findById(ID var1), findById(ID var1) save(S var1), save(S var1)
		/mentor/swipe/ajax/exit	getSwipeExitDetails(List<String> mentorId, String feedback, String userId, String attendance)	findById(ID var1), findById(ID var1), findCurrentUser(@Param("user") User user, @Param("today") Date today),(@Param("user") User user, @Param("today") Date today),

				findCurrentUser(@Param("user") User user, @Param("today") Date today),(@Param("user") User user, @Param("today") Date today),
	Calendar & Swipe Screen	/mentor/swipe/ajax/register	getSwipeRegisterDetails(String username, String password, String cardId, String sessionTimeSlotId)	findById(ID var1), save(S var1), findByUsername(S var1), findByName(S var1), save(S var1)
	Absence	/mentor/absence/market	getAbsenceDetails()	findByUsername(String username), getAbsencesByMentorUserName(@Param("netid") String netid), getAbsencesByMentorUserNameAndToday(@Param("netid") String netid)
		/mentor/absence/create	getMentorAbsenceFormResultDTO()	getShiftAssignmentByMentorUserName(@Param("netid") String netid)
		/mentor/absence/create/submit	submitAbsence(String date, String time, String reason, String absenceId)	getShiftAssignmentByDateAndTime(@Param("netid") String netid, @Param("date") Date date, @Param("time") Date time), save(S var1), save(S var1), findById(ID var1), save(S var1)
		/mentor/absence/cancel/{absenceId}	cancelAbsence(UUID absenceId)	findById(ID var1), findById(ID var1), save(S var1), deleteById(ID var1)
		/mentor/absence/update/{absenceId}	updateAbsence(UUID absenceId)	findById(ID var1),

				getShiftAssignmentByMentorUserName(@Param("netid") String netid)
		/mentor/absence/claim/{absenceId}	claimShift(UUID absenceId)	findById(ID var1), save(S var1), save(S var1)
		/mentor/absence/shift	getShiftsByDate(String date)	
	Grades	/mentor/session/history	getScheduledSession()	getScheduledSessions(@Param("today") Date today)
			getQuizzes()	getQuizzes(@Param("today") Date today)
		/mentor/session/grades/{Id}	getDetailsForScheduledSessionOrQuiz(ModelMap model, UUID id)	getQuizAttendanceForGrades(@Param("quizId") UUID quizId), findSessionTimeSlotBySessionId(@Param("id") UUID id)
		/mentor/attend/{Id}	getDetailsForAttendance(UUID id)	getAttendanceGrade(@Param("sessionid") UUID sessionid), getAttendanceGrade(@Param("sessionid") UUID sessionid)
		/mentor/ajax/session/grades	updateGrade(int grade, UUID attendance)	findById(ID var1), save(S var1)
		/mentor/attend/submit/{Id}	markStudentAttendance(UUID sessionId, String timeSlot, String netId, String grade)	findById(ID var1), findByUsername(String username), save(S var1), findByUsername(String username), findById(ID var1), findById(ID var1)
			getDetailsForScheduledSessionOrQuiz(ModelMap model, UUID id)	getQuizAttendanceForGrades(@Param("quizId") UUID quizId), findSessionTimeSlotBySessionId(@Param("id") UUID id)
	Timesheet Report	/mentor/timesheet /mentor/timesheet/{dt}	getTimeSheetofUser()	getTimeSheetForUser(@Param("netid") String netid)

			getCurrentStartDateofTheWeek()	
	Display	/mentor/display	getMentorDetails()	getSessionDisplayDetails(@Param("date") Date date), getSpecialtiesDisplayDetails(@Param("user") User user)
			getStudentDetails()	getStudentDisplayDetails(@Param("date") Date date)
			getSessionDetails()	getSessionDisplayDetails(@Param("firstDate") Date firstDate, @Param("lastDate") Date lastDate)
			getQuizzDetails()	getQuizDisplayDetails(@Param("now") Date now)
			getShiftLeaderDetails()	getShiftLeaderDisplayDetails(@Param("now") Date now, @Param("todayWeekday") Integer todayWeekday)
Instructor		/instructor/course/section	getUser()	findByUsername(String username)
			getInstructorSections(User instructor)	findAllSectionsByInstructorUsername(@Param("user") User user)
		/instructor/course/section/view/{id}	getSectionById(String id)	findById(ID var1)
			findUserByNetId(String[] data)	findByUsername(String username), save(S var1)
			enrollUserToSection(Section section, List<User> users)	save(S var1)
		/instructor/course/section/view/{id}	getRoster(UUID id)	findStudentsBySectionId(@Param("sec_id") UUID id)
			getSectionById(String id)	findById(ID var1)
		/instructor/session/schedule	getSessionSchedule(String userId)	findSectionsByInstructorUsername(@Param("netid") String netid)
		/instructor/session/schedule/session/view/{id}	getSession(UUID id)	findById(ID var1)
		/instructor/session/schedule/session/view/{sessionId}/download/{fileId}	getFileById(UUID id)	findById(ID var1)
			getFileMetaDataByKey(File file, String key)	findByKey(@Param("file") File file, @Param("key") String key)
		/instructor/session/request	getUser()	findByUsername(String username)

			findSessionsByUser(User u)	findRequestsByUser(@Param("user") User user)
		/instructor/session/request/create	getInstructorCurrentSections(String netId)	findSectionsByInstructorUsername(@Param("netid") String netid)
		/instructor/session/request/create	getSessionTypeByName(String name)	findSessionTypeByName(@Param("name") String name)
			addSessionRequest(InstructorRequestSessionDTO requestSessionDTO, User instructor, SessionType type)	save(S var1)
		/instructor/session/request/view/{id}	findByRequestId(UUID id)	findById(ID var1)
		/instructor/session/request/view/{requestId}/download/{fileId}	getFileById(UUID id)	findById(ID var1)
			getFileMetaDataByKey(File file, String key)	findByKey(@Param("file") File file, @Param("key") String key)
			getFileMetaDataByKey(File file, String key)	findByKey(@Param("file") File file, @Param("key") String key)
		/instructor/session/request/edit/{id}	findByRequestId(UUID id)	findById(ID var1)
			getInstructorCurrentSections(String netId)	findSectionsByInstructorUsername(@Param("netid") String netid)
		/instructor/session/request/edit/{id}	getUser()	findByUsername(String username)
			findByRequestId(UUID id)	findById(ID var1)
			findSessionTypeById(UUID id)	findById(ID var1)
			getSectionsFromIdList(List<String> sectionList)	findSectionsByIdList(@Param("sectionIds") List<UUID> sectionIds)
			saveFile(File file)	save(S var1)
			updateSessionRequest(Request request)	save(S var1)
		/instructor/session/history	getAllGradedSessionByInstructor(UserDTO userDTO)	findAllGradedSessionsByInstructorOrderByStartTime(@Param("netId") String netId, @Param("today") Date today)
			getAllGradedQuizByInstructor(UserDTO userDTO)	findAllGradedQuizByInstructorOrderByStartTime(@Param("netId") String netId, @Param("today") Date today)
		/instructor/session/grades/session/{scheduleSessionId	getSessionInfoById (UUID sessionId)	findAllById(Id var1)
			getAllAttendanceBySession(ScheduledSession session)	findAllBySession(@Param("session") ScheduledSession session)

		/instructor/session/grades/quiz/{quizId}	getQuizInfoById(UUID quizId)	findAllById(Id var1)
			getQuizAttendanceByQuiz(Quiz quiz)	findAllAttendanceById(@Param("quizSession") Quiz quizSession)
		/instructor/session/grades/session/session/grades/download/{sessionId}	getSessionInfoById (UUID sessionId) (UUID sessionId)	findAllById(UUID sessionId)(UUID sessionId)
			getAllAttendanceBySession(ScheduledSession session)(ScheduledSession session)	findAllBySession(@Param("session") ScheduledSession session)(@Param("session") ScheduledSession session)
		/instructor/session/grades/quiz/session/grades/download/{quizId}	getQuizInfoById(UUID quizId)(UUID quizId)	findAllById(UUID quizId)(UUID quizId)
			getQuizAttendanceByQuiz(Quiz quiz)(Quiz quiz)	findAllAttendanceById(@Param("quizSession") Quiz quizSession)(@Param("quizSession") Quiz quizSession)
Admin				
	Home	/admin/home	getHomeScreenDetails(ModelMap model)	findByDay(S var1), findByTime(S var1, S var2)
	Announcements	/admin/announcement	getAnnouncement()	findAll()
		/admin/announcement/create	getRole()	findAll()
		/admin/announcement/create/submit	submitAnnouncement(adminAnnouncementResultDTO)	Save(newAnnouncement)
		/admin/announcement/edit/{id}	getAnnouncementDetails(adminAnnouncementResultDTO)	FindById(announcementId)
		/admin/announcement/update/{announcementId}	updateAnnouncement(adminAnnouncementResultDTO)	Save(announcement)
		/admin/announcement/delete/announcementId		
	System Set Up			
	IPs	/admin/ip	getIpDetails(model)	findAll()
		/admin /ip/create	-	-
		/admin /ip/create/submit	submitIpDetails()	findByAddress(Long address), save()
		/admin /ip/edit/{ipId}	getIpAddressDetails(String ipId)	findById(UUID id)
		/admin /ip/update/{ipId}	updateIpDetails(AdminIpResultDTO adminIpResultDTO)	findById(UUID id)
		/admin ip/delete/{ipId}	deleteIp(UUID ipId)	findById(UUID id)

	Rooms	/room/list	getListofRooms()	findAll()
		/room/create	-	-
		/room/create/submit	submitRoomDetails(AdminRoomResultDTO adminRoomResultDTO)	save()
		/room/edit/{roomId}	updateRoomDetails(AdminRoomResultDTO adminRoomResultDTO)	findById(UUID id), save()
		/room/delete/{roomId}	deleteRoom(UUID roomId)	findById(UUID id)
	Semesters	/admin/semester	getSemesterList()	findSemesterList()
		/admin/semester/create	getSeasonsNames()	
		/admin/semester/edit/{semesterID}	saveSemester(season, year, startdate, enddate, isSupported)	findSemesterList() setSemesterNotActive(semester1)
			buildSemesterModel(newSemester,season,year,startDate,endDate,isActive)	getSemesterFromSeasonandYear(season,Integer.parseInt(year))
		/admin/semester/update/{semesterID}	saveEditedSemester(semesterID, season, year, startdate, enddate, isSupported)	getSemester(UUID.*fromString*(semesterID)) findSemesterList() getSemesterFromSeasonandYear(season,Integer.*parseInt*(year))
		/admin/semester/create/submit	saveSemester(season, year, startdate, enddate, isSupported)	findSemesterList() getSemesterFromSeasonandYear(season,Integer.*parseInt*(year))
		/admin/semester/delete/{semesterId}	deleteSemester(semesterId)	delete(semester)
	Departments	/admin/department	getDepartmentDetailsList()	findAll()
		/admin/department/create	-	-
		/admin//department/create/submit	submitDepartmentDetails(AdminDepartmentsResultDTO adminDepartmentsResultDTO)	findByUsername(String username)
		/admin/department/edit/{departmentId}	getDepartmentDetails(final AdminDepartmentsResultDTO adminDepartmentsResultDTO)	findById(ID var1)
		/admin/department/update/{departmentId}	updateDepartmentDetails(AdminDepartmentsResultDTO adminDepartmentsResultDTO)	findByUsername(String username), save(S var1)

		/admin/department/delete/{departmentId}	deleteDepartment(UUID departmentId)	findById(ID var1), delete(T var1)
	Courses	/admin/course	getSupportedCourseDetails()	findCourseBySupported(boolean supported)
			getUnsupportedCourseDetails()	findCourseBySupported(boolean supported)
		/admin/course/create	getDepartmentsForNewCourse()	findAllByOrderByName()
		/admin/course/create/submit	submitCourseDetails(AdminCourseDetailsResultDTO adminCourseDetailsResultDTO)	findDepartmentByName(String name), findByUsername(String username), save(S var1)
		/admin/course/edit/{courseId}	getCourseDetails(final AdminCourseDetailsResultDTO adminCourseDetailsResultDTO)	findById(ID var1)
			getDepartmentsForNewCourse()	findAllByOrderByName()
		/admin /course/update/{courseId}	updateCourseDetails(AdminCourseDetailsResultDTO adminCourseDetailsResultDTO)	findById(ID var1), findByUsername(String username), findDepartmentByName(String name), save(S var1)
		/admin/course/delete/{courseId}	deleteCourse(UUID courseId)	findById(ID var1), delete(T var1)
	Sections	/admin/section	getRawSectionList()	getSectionList()

		/admin/section/create	getCourseList()
			getSemesterList()
			getInstructors()
				findCourseBySupported(true),
				
				findSemesterList(),
				
				getUserByRole(role.getId())

		/admin/section/create/submit	buildSection(cnumber, number, semester, instructor, tassistant, adminNotes)	getCourse(UUID.fromString(cnumberID)), getSemester(UUID.fromString(semesterID)), getUser(UUID.fromString(instrID)),

				getUser(defaultUsernameServ ice.getUsername())
		/admin/section/saveAllRoster	getSectionsList(model)	
		/admin/section/{sectionId}/sa veRoster	getSection(sectionId) buildSection(cnumber, number, semester, instructor, tassistant, adminNotes)	getCourse(UUID.fromString(cn umberID)), getSemester(UUID.fromString(semesterID)), getUser(UUID.fromString(instr ID)), getUser(defaultUsernameServic e.getUsername())
		/admin/section/edit/{sectionId }	getSection(sectionId) getCourseList() getSemesterList() getInstructors()	getSectionFromId(UUID.fromS tring(id)), findCourseBySupported(true), getSectionFromId(UUID.fromS tring(id)), getUserByRole(role.getId())
		/admin/section/saveAllRoster	getAllSectionIDs(file)	getSectionList(), getUser(filenetId), getUser(filenetId), getCourseID(fileCourseNumber), getSemesterID(season, year), getSectionFromNumber(section Id)
		/admin/section/roster/{section Id}	getSection(sectionId))	
		/admin/section/{sectionId}/sa veRoster	saveRoster(sectionId, file) getSection(sectionId)	getSectionFromId(UUID.fromS tring(sectionId)), getSectionFromId(UUID.fromS tring(id))
		/admin/section/update/{sectio nId}	saveEditedSection(section Id, cnumber, number, semester, instructor, tassistant, adminNotes)	getSectionFromId(UUID.fromS tring(sectionId)), getCourse(UUID.fromString(co useID)),

				getSemester(UUID.fromString(semesterID)), getUser(UUID.fromString(IntsID))

		/admin/section/delete/{sectionId}	deleteSection(sectionId)	getSectionFromId(UUID.fromString(sectionID))
				delete(section)

	Operation Hours	/admin/hours	getOperationHours()	findAll()
		/admin/edit/hours/{id}	editOperationHours(UUID id)	findById(id)

		/admin/edit/hours/submit/{hoursId}	saveOperationHours(AdminOperationHoursResultDTO adminOperationHoursResultDTO)	findById(adminOperationHoursResultDTO.getOperationHoursId())
				save(optHours)

	Users			
	User	/admin/users/user		
		/admin/users/user/create	getNewUserForm()	findAll()
		/admin/users/user/create/submit	createNewUser(String userName, String firstName, String lastName, String cardId, String scanCode, String[] roleSelected)(String userName, String firstName, String lastName, String cardId, String scanCode, String[] roleSelected)	findById(ID var1)
				findByUsername(String username)
				findByCardId(String scancode)
				findByScancode(String scanCode)
				save(S var1)
				findAll()
		/admin/users/user/update/{userId}userId}	getUpdateUserForm(UUID userId)(UUID userId)	findAll()
				findById(ID var1)
		/admin/users/user/update/submit/{userId}userId}	updateCreatedUser(UUID userId, String userName, String firstName, String lastName, String cardId, String scanCode, String[] roleSelected)(UUID userId, String userName, String firstName, String lastName, String cardId, String	findById(ID var1)

			scanCode, String[] roleSelected)	
				findByUsername(String username)
				findByCardId(String scancode)
				findByScancode(String scanCode)
				findById(ID var1)
				save(S var1)
				findAll()
		/admin/users/user/queryByPagequeryByPage	queryUsersByPage(DataTablesInput input)(DataTablesInput input)	findUsersByPage2(@Param("word") String word, Pageable pageable)
		/admin/users/user/delete/{userId}userId}	deleteUser(UUID userId)(UUID userId)	findById(ID var1)(ID var1)
				deleteUser(@Param("userId") UUID userId)(@Param("userId") UUID userId)
	Roles	/admin/user/role/	getRoleResult())	findAll(););

		/admin/user/role/create/submit/	createRole(ModelMap model, String rolename, String description)	findByName(String rolename);
				save(newRole);

		/admin/user/role/edit/submit/{roleID}/	editRole(ModelMap model, UUID roleID, String rolename, String description)	findByName(String rolename);
				findById(UUID roleID);
				findByName(String rolename);
				save(Role role);

		/admin/user/role/edit/{roleID}/	getRoleByID(UUID roleID)	findById(UUID roleID)

		/admin/user/role/remove/{roleID}/	removeRole(UUID roleID)	findById(UUID roleID)
				getUsersByRole(@Param("rolename") String rolename);
				delete(Role role);

	Sessions			

	Requests	/admin/session/request/	getSessionTable()	getRequestsByStatus2(@Param("status") Request.RequestStatus status);
				save(Scheduledsession ss);
				save(Quiz q);

		/admin/session/request/edit/{sessionId}/{sessionType}	getSessionNewRequestDTO(UUID sessionID, String type)	findById(UUID id);
				findAll();

		/admin/session/edit/{sessionId}/{sessionType}	getSessionForm(UUID sessionID, String type)	findById(UUID id);
				findAll();

		/admin/session/edit/uploadFile/{sessionId}/{sessionType}	uploadFileToSession(File f, String sessionId, String sessionType)	findById(UUID id);
				save(Scheduledsession ss);
				save(Quiz q);

		/admin/session/edit/submit/{sessionId}/{sessionType}	updateSession(AdminSessionUpdatedDTO adminSessionUpdatedDTO, Map<String,String> allRequestParams)	findAll();
				findById(UUID id);
				save(Scheduledsession ss);
				save(Quiz q);
				save(QuizTimeSlot qts);

		/admin/session/create/{requestId}/{sessionType}	approveRequestSession(UUID id, String type)
			getSessionForm(UUID sessionID, String type)
				findById(UUID id);
				save(Scheduledsession ss);
				save(Quiz q);
				save(Request r);
				save(QuizTimeSlot qts);
				findAll();

		/admin/session/deny/{sessionId}/{sessionType}	denyRequestSession(UUID id, String type)	save(Request r);
				findById(UUID id);

		/admin/session/grades/{sessionId}/{sessionType}	getDetailsForScheduledSessionOrQuiz(ModelMap model, UUID id, String ssType)	findById(UUID id);
				getQuizAttendanceForGrades(@Param("quizId") UUID quizId)

		/admin/ajax/session/grades	updateGrade(int grade, UUID attendance)	findById(UUID id);
				save(SessionAttendance sa);

		/admin/session/attend/{Id}/{sessionType}	getDetailsForAttendance(UUID id, String ssType)	findById(UUID id);

		/admin/session/attend/submit/{Id}/{sessionType}	markStudentAttendance(UUID sessionId, String timeSlot, String netId, String grade, String ssType)	findById(UUID id);
				save(Quiz quiz);

	Staffing			
	Schedule	/admin/ schedule/calendar	getCalendarScheduleDetails()	findAll(), findUserByRolesName(String name), findByActive(boolean b), findByShowOnCalendar(boolean showOnCalendar)
		/admin/schedule/ajax/shift/leader	updateShiftLeader(RequestObjectSchedule requestObjectSchedule)	findById(ID var1), findById(ID var1), saveAndFlush(S var1)
		/admin/schedule/ajax/shift/remove	removeMentorFromShift(RequestObjectShift requestObjectShift)	findById(ID var1), findById(ID var1), findById(ID var1), saveAndFlush(S var1)

		/admin/schedule/ajax/shift	updateShift(RequestObject Shift requestObjectShift)	findById(ID var1), findById(ID var1), findById(ID var1), saveAndFlush(S var1)
		/admin/subject/ajax/color	updateSubjectColor(Reques tObjectColor requestObject)	findById(ID var1), saveAndFlush(S var1)
		/admin/schedule	getScheduleTimes()	findByActive(boolean b), findBySchedule(Schedule schedule)
			getScheduleRooms()	findByActive(boolean active)
		/admin/schedule/increase	updateIncMaxMentors(Req uestObjectMaxMentor requestObjectMaxMentors)	findById(ID var1), saveAndFlush(S var1)
		/admin/schedule/decrease	updateDecMaxMentors(Re questObjectMaxMentor requestObjectMaxMentors)	findById(ID var1), saveAndFlush(S var1)
		/admin/schedule/create	createShift(AdminCreateSh iftResultDTO adminCreateShiftResultDT O)	findByActive(boolean b), findById(ID var1), findByShowOnCalendar(bool ean showOnCalendar), saveAndFlush(S var1)
		/admin/schedule/update	getUpdateDetails()	findByActive(boolean b), findByScheduleAndShiftAnd Date(Schedule schedule, Shift shift, Date date), save(S var1), findBySubstitute(ShiftAssign ment currentAssignment), deleteSingleQuery(@Param("i d") UUID id), saveAndFlush(S var1)
	Timsheets	/admin/schedule/timesheets	getMentor()	findUserByRolesName("ment or")
		/admin/schedule/timesheets/re port	getTimesheet(adminTimesh eetResultDTO)	GetTimesheetForMentor(User userId ,Date startDate, Date endDate)
	Absences	/admin/schedule/absences	getAbsenceDetails()	getAbsenceDetails()
	Mentors	/admin/mentor	getMentorDetails()	findUserByRolesName("ment or")

	Reports	/admin/report	getAllSectionsCurrentSemester()	findAllSectionActiveSmester(),
			getAllSessionsCurrentSemester()	findAllSessionJoinSectionActiveSemester()
		/admin/report/submit	getAllSectionsCurrentSemester()	findAllSectionActiveSmester()
			getAllSessionsCurrentSemester()	findAllSessionJoinSectionActiveSemester()
			getReportForSession (Section section, ScheduledSession session)	
			getWalkInAttendanceBySection(Section section)	getAllByCourse(S var1)
		/admin/report/submit/report/download/session/{section_id}/{session_id}	getAllSectionsCurrentSemester()	findAllSectionActiveSmester(),
			getAllSessionsCurrentSemester()	findAllSessionJoinSectionActiveSemester()
			getReportForSession (Section section, ScheduledSession session)	
		/admin/report/submit/report/download/walk_in/{section_id}	getAllSectionsCurrentSemester()	findAllSectionActiveSmester()
			getWalkInAttendanceBySection(Section section)	getAllByCourse(S var1)
	Swipes	/admin/swipes	getSwipes()	findAll()

5. Software Risk Issues

There are several parts of the project that are not within the control of the new CSMC but have direct impacts on the process and must be checked as well:

- The ability to restart the application in the middle of the process is a critical factor to application reliability.
- Database security and access must be defined and verified

6. Features to be Tested

Below are features which are implemented and tested.

- Student view:
  - Sign-in up for a session
  - Register
  - Unregister
  - Grades
- Mentor View
  - Staff schedule page
  - Swipe screen for walk-in

    - Session details view Grades
    - Absences
    - Timesheet reports
- Instructor view:
    - Request session
    - Review session results (grades)
    - Reports
    - Course
    - Session Schedule
- Admin view:
    - Home
    - Announcements
    - System setup
        - IPs
        - Rooms
        - Departments
        - Sections
        - Semesters
        - Courses
        - Operation Hours
    - Users
        - Users
        - Roles
    - Sessions
        - Requests
    - Reports
    - Staff
        - Schedule
        - Mentors
        - Timesheets
        - Absences
    - Swipes

7. Features not to be Tested

Below are the features that are not completed and tested:

- Pre-login Home Page
- Mentor View
    - Display [asset function using which images/CSS are getting displayed] [CSS not responsive]
    - Calendar [SSO validation for new user registration for Register flow] [No card form flow] [asset function using which images/CSS are getting displayed] [Materials download flow]
    - Swipe Screen [SSO validation for new user registration for Register flow] [asset function using which images/CSS are getting displayed]
- Admin View
    - Users

        - Users groups
    - Sessions
        - Holidays
        - Calendar

8. Approach

8.1 Testing Levels

- The testing for the CSMC system will consist of Unit, System/Integration (combined) and Acceptance test levels. It is hoped that there will be at least one full time independent test person for system/integration testing. However, due to limited conditions; most testing will be done by the developers in the class.
- UNIT Testing will be done by the developer and will be approved after weekly meeting with CSMC stakeholders. Proof of unit testing (test case list, sample output, data printouts, defect information) must be provided by the programmer to the entire group before unit testing will be accepted and the code is merge to the main branch. All unit test information will also be provided to the test person.
- SYSTEM/INTEGRATION Testing will be performed by the CSMC staffs and development team leader with assistance from the individual developers as required. No specific test tools are available for this project. Programs will enter into System/Integration test after all critical defects have been corrected.
- ACCEPTANCE Testing will be performed by the actual end users with the assistance of the CSMC director. The acceptance test will be done and demo the final presentation.
- Programs will enter Acceptance test after all critical and major defects have been corrected. Prior to final completion of acceptance testing all open critical and major defects MUST be corrected and verified by the User representative.

8. 2 Test tools.

- IDE: IntelliJ or Eclipse – for running code and debugging
- MAMP – for generating local host

8.3 Meetings

The entire team will meet once a week with the CSMC stakeholders to evaluate progress to date and to identify error trends and problems as early as possible. Other than that, the students will meet on another day to discuss the progress, divide that tasks and resolve any problems faced that can be resolved by the help of other members. Additional meetings can be called as required for emergency situations. Meeting is created on Zoom App or Microsoft Team App.

9. Item Pass/Fail Criteria

The CSMC stakeholders will provide different test cases from the current CSMC website for individual features which will be matched with the developed website to determine if the feature in the new system is successfully developed.

10. Suspension Criteria and Resumption Requirements

10.1 Suspension Criteria

- The application build contains some serious technical defects that can limit testing progress.

- Significant changes in the requirements.
- Sandbox server hosting the database becomes unavailable.

10.2 Resumption Requirements

Resumption will occur when the problem(s) that caused the suspension have been resolved.

11 Test Deliverables

- Acceptance test plan (in final presentation)
- System/Integration test plan (in final presentation and weekly meeting)
- Unit test plans/turnover documentation (in weekly meeting)
- Video of usage (final submission)
- Report and progress (weekly meeting)

12. Remaining Test Tasks

- All implemented features are tested, there is no remaining test tasks
- For the not implemented features, the developer who will be working on that will be responsible for testing tasks.

13. Environmental Needs

The following elements are required to support the overall testing effort at all levels of the project:

- MacOS or Windows operation system.
- IDE: IntelliJ or Eclipse
- Configuration to connect to Sandbox server database
- Browser

14. Staffing and Training Needs

In order to provide complete and proper testing the following areas need to be addressed in terms of training.

- A. The developers and tester(s) will need to be trained on the basic operations of the CSMC interface. Prior to final acceptance of the project the operations staff will also require complete training on using the system in different roles.
- B. The administration staff will require training on the entire system.
- C. At least one developer and operations staff member needs to be trained on the installation and control of the PC based for the code.

15. Responsibilities

	Team Dev	CSMC staff	CSMC director	Instructors
Acceptance test	x	x	x	x
System/Integration test	x	x	x	
Unit test plans/turnover documentation	x	x		
Unit test plans/turnover documentation	x			
Video of usage	x			
Report and progress	x			

16. Schedule

Task	Time
Student View test (unit test and integration test)	February 2020
Mentor View test (unit test and integration test)	March 2020
Instructor View test (unit test and integration test)	April 2020
Admin View test (unit test and integration test)	April 2020
All view (System/Integration test)	May 2020
Acceptance test	May 2020
Report of progress	Weekly
Final presentation	05/06/2020
Video of usage and Final documentation	05/08/2020

17. Planning Risks and Contingencies

- Lack of testing specialist.
- As a result of this staff shortage there may be delays in getting staff to review appropriate documents and to participate in the Acceptance test process.
- Future deploying and testing may have problems since all Dev team will leave after May 2020.