Design Inspection, Code Inspection, Unit Testing

Product	Tindents
Date	February 7, 2019
Authors	Rahul Balla, Andre Chang, Sriharsha Valluripalli, Meriem Bounab, Brent Singh Sharkey

Design Inspection

Defect #	Description	Severity (1 high - 5 low)	How Corrected
1	File: SwipingViewController.swift After swiping right or left to either like or dislike a tutor, the image of the next tutor would come back in the rotated position of how the first tutor exited the screen.	3	The angle at which the card should be reset to get to the original position was calculated and implemented in the function which fetches the next tutor
2	File: ViewController.swift Program would redirect you to the main screen when pressing login even if no login information was entered.	1	*currently fixing
3	File: matchesTableViewController.swift After pressing on a tableview cell of one of your matches, you are redirected to the match profile page without a back reference to the list of matches.	4	Added a navigation view controller which allowed for pages to be pushed and popped from the stack of view controllers

4	File: Main.Storyboard There were no constraints between different UI components of the view controllers and that is why they got misplaced when simulating in different emulators	3	Fixed by setting auto layout constraints between UI elements
5	File: Main.Storyboard After the user logs in, they are redirected to the Settings tab instead of the Feed tab to search for tutors	4	Replaced the starting tab to the Feed tab in the code for tab control
6	File: matchesTableViewCell.swift When the user inputs a large enough enough bio, the information will be cut off in the list of matches	5	*currently fixing
7	File: Main.Storyboard Labels of the different filters were overlapping over each other so that we can't read them anymore	3	Made the label sizes smaller and changed the constraints of the different components

Code Inspection

Defect #	Description	Old Code	Severity (1 - 5) 1 = High 5 = Low	How Corrected	New Code

1	File: signupViewController
	
	Trying to connect json file to a http request causes an "Error thrown from here are not handled"
		request.httpBody = JSONSerialization.data(withJSONObject: params, options: [])	2	Fixed by encasing the statement with a try catch statement.	**do** {
					try
					request.httpBody = JSONSerialization.data(withJSONObject: params, options: [])
					
					print(request.httpBody)
					} **catch** {
					print("???")
					}

2	File: signupViewController Trying to send a POST request of json file format to the server returns an error	**var** request = NSMutableURLRequest(url: NSURL(string: "http://127.0.0.1:5000/createAccount")! **as** URL)	3	Fixed by specifying json format for the POST request	var request = NSMutableURLRequest(url: NSURL(string: "http://127.0.0.1:5000/createAccount")! as URL) request.addValue("application/json", forHTTPHeaderField: "Content-Type")

3	File: SwipingViewController
	
	When swiping the card it would always return to the center of the screen after the user releases their finger. This was because the code for swiping the card off the screen was outside the if condition for when
		if sender.state == .ended {
		
		UIView.animate(withDuration: 0.2) {
		card.center = **self**.view.center
		
		self.card.transform = .identity
		}
		}
		
		if (card.center.x < 75) {
		
		UIView.animate(withDuration: 0.3) {
		card.center = CGPoint(x:
			2	It was fixed by by shifting the code into the if statement when the user state had ended	if (card.center.x < 75) {
					
					UIView.animate(withDuration: 0.3) {
					card.center = CGPoint(x: card.center.x - 200, y: card.center.y + 75)
					card.alpha = 0
					}
					return
					}
					UIView.animate(withDuration: 0.2) {
					card.center = self.view.center
					
					self.card.transform = .identity
					}

	the user state had ended.	card.center.x - 200, y: card.center.y + 75) card.alpha = 0 } **return** }			

4	File: Tutor.swift
	
	Getting each tutor's information from the server and parsing it within the same view controller was difficult
		No old code	2	A separate data structure was created to store the tutor's information like their name, subjects they will teach and other things. This will make it easier to store and parse the tutor's data	class Tutor {
					
					var name: String?
					var subjects: [String]?
					var tutorEmail: String?
					var rating: String?
					var picture: UIImage?
					var description: String?
					
					init (dictionary: [String: Any]) {
					name = dictionary["name"] as? String ?? "Creepy Tutor"
					subjects = dictionary["subjects"] as? [String] ?? nil
					tutorEmail = dictionary["tutorEmail"] as? String ?? "No email"
					rating = dictionary["rating"] as? String ?? "No Rating"
					picture = dictionary["picture"] as? UIImage?? nil
					description = dictionary["description"] as? String ?? "No Description"
					}
					}

5	settingsViewController.swift
	
	The program could not correctly recognize when a switch was pressed
		@IBOutlet weak var mathSwitch: UISwitch!	2	To fix this I had to change the switch from an outlet to an action so that the program can be listening and trigger on the switch changing action	**@IBAction func** mathSwitch(sender: UISwitch) {
					
					if(sender.isOn==**true**){
					
					mathSwitchOn=**true**;
					}
					else{
					
					mathSwitchOn=**false**;
					}
					}

Unit Testing Defects

Unit Testing Defects

≈	Component Being Tested	Description of Testing Results	Severity	How Corrected
1	Login Screen	When trying to login without entering any input in the username and password field, the server spit out an error	1	We modified the swift code for the login screen so that the input fields are not empty. When a fields are empty, the app would display an alert box prompting to enter a valid input.
2	Sign Up Screen	Similar to the login screen, the server spit out an error where there were empty fields in the sign up screen	2	We modified the swift code for the sign up screen so that the input fields are not empty. When a fields are empty, the app would display an alert box prompting to enters valid input.
3	Sign Up Screen	The confirm password screen was not checking if it was same as the password entered.	2	We modified the swift code for sign up screen so that when the user enters input in both the password fields and submits changes, they would be prompted that “passwords don’t match” if there they enter different inputs in password and confirm password fields
4	Matches Screen	The matches screen was sometimes showing two copies of the same person	3	To fix this, we modified the backend code to make sure that a student can match with a tutor only once. We checked if the table storing the matches already had the student-tutor pair. If the pair exists, then we don’t add the pair again to the table. We modified so that there are no duplicates in the matches table.
5	Navigation Bar	The settings button on the navbar at the bottom of the page would navigate to the user’s profile and not the settings page.	2	To fix this, we made sure that the buttons are routed to their appropriate pages.

6	Filter Switches	When a user would sign in, all of the filters would automatically be turned on as a default	2	To fix this, we changed the settings of each switch so that when initialized, they would be turned off as a default.
7	Tutor Info Screen	The input field where the tutor can enter his/her hourly charge was accepting negative values as inputs.	2	To fix this, we modified the swift code so that the page would give an alert popup when the user enters a -ve value.
≈	Component Being Tested	Description of Testing Results	Severity	How Corrected
1	Backend Database	Database refused to connect to RDBMS	5	Altered AWS security permissions to allow incoming connections and not void-out outgoing comissions.uu