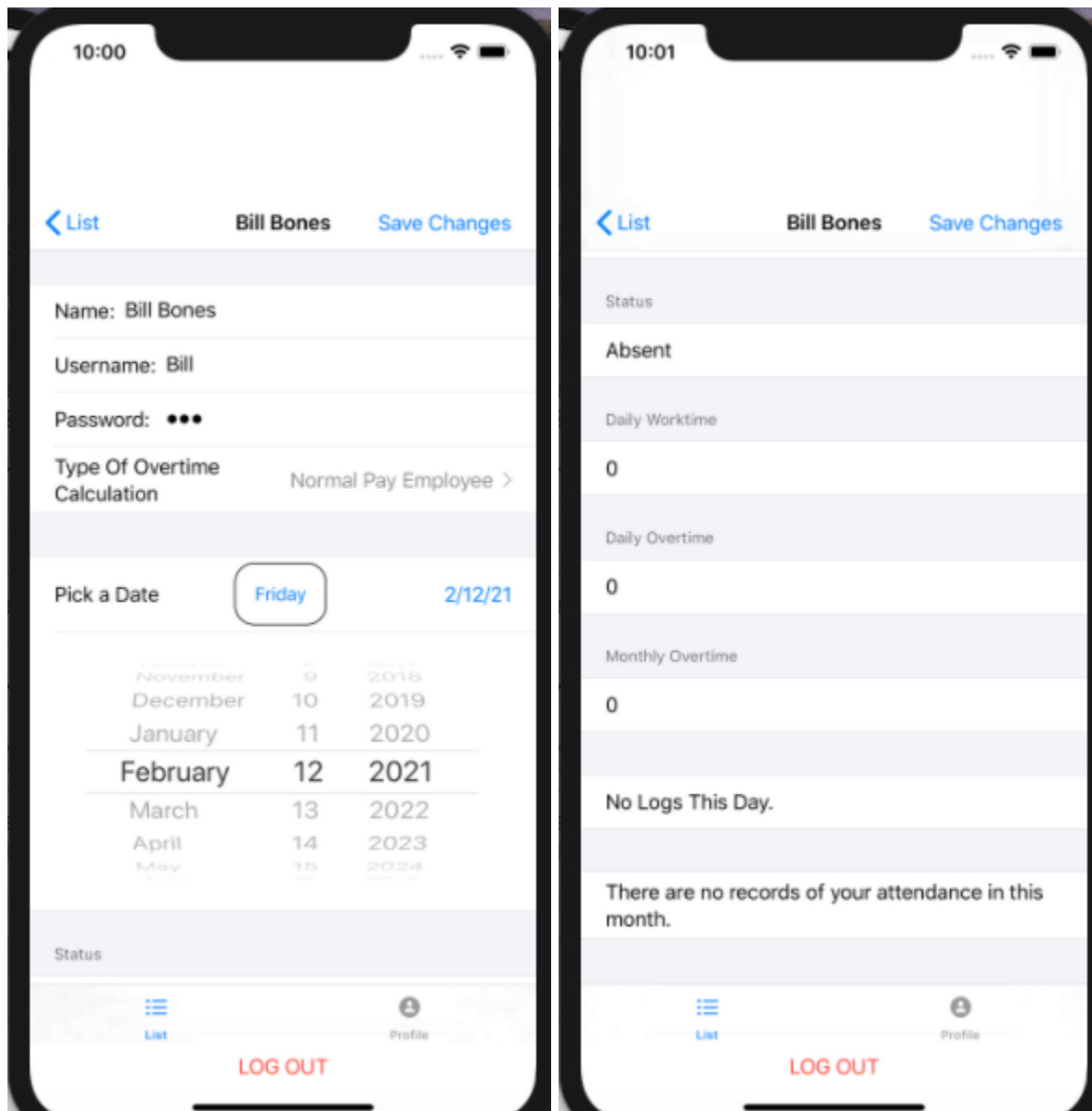


## Employee information view page



Day and Status continued

	October 2020
11	SUNDAY
12	ABSENT
13	ABSENT
14	ABSENT
15	ABSENT
16	ABSENT
17	ABSENT
18	SUNDAY
19	ABSENT
20	ABSENT
21	ABSENT
22	ABSENT
23	ABSENT
24	ABSENT
19	ABSENT
20	ABSENT
21	ABSENT
22	ABSENT
23	ABSENT
24	ABSENT
25	SUNDAY
26	ABSENT
27	ABSENT
28	ABSENT
29	ABSENT
30	ABSENT
31	ABSENT

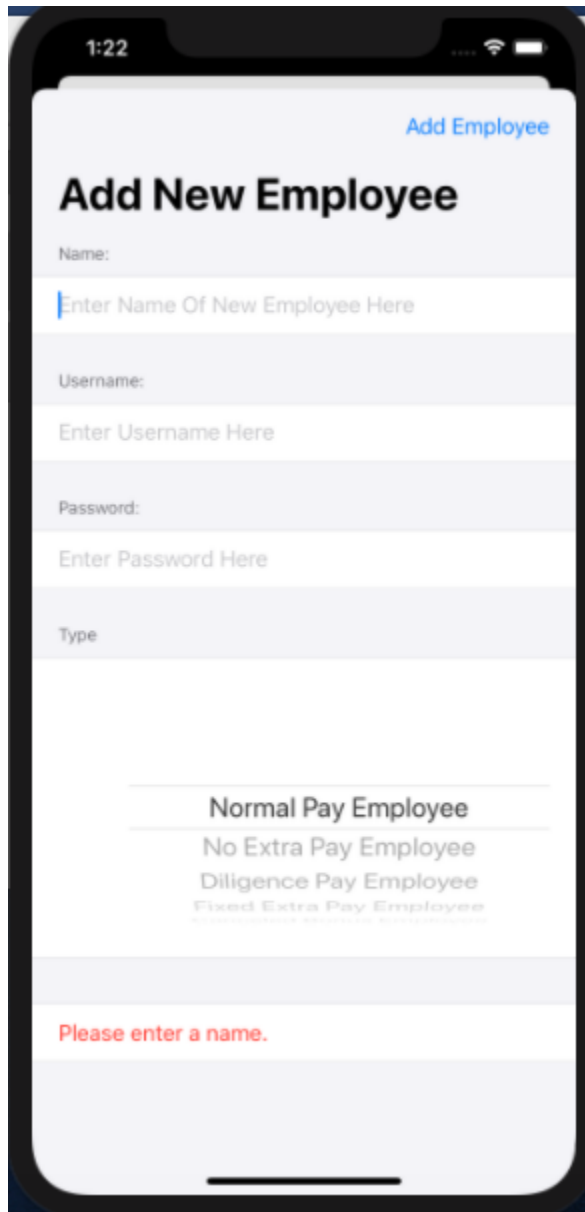
## ClientAccountView Struct

```
12 struct ClientAccountView: View {
13     var client: Client //client
14     @Environment(\.managedObjectContext) var moc //managed object context
15     @FetchRequest(entity: Month.entity(), sortDescriptors: []) var months: FetchedResults<Month> //all months
16     @State private var name: String //basic client information
17     @State private var email: String
18     @State private var username: String
19     @State private var password: String
20     @State private var isSaved = false
21     @State private var dateToSendAsJson = Date() //the date that is selected by the client to send as JSON.
22     private var monthAsText: String {
23         "\((DateFormatter().monthSymbols[(components.month ?? 1)-1]) \((components.year ?? 2021)"
24     } //UI text
25     private var components: DateComponents {
26         Calendar.current.dateComponents([.month, .year], from: dateToSendAsJson) //month and year components of the date
27     }
28     init(client: Client) {
29         self.client = client
30         _name = State(initialValue: client.wrappedName)
31         _email = State(initialValue: client.wrappedEmail)
32         _username = State(initialValue: client.wrappedUsername)
33         _password = State(initialValue: client.wrappedPassword)
34     }
35     @State var alertShowing = false
36
37     var body: some View {
38         NavigationView {
39             Form {
40                 //editable fields, formatted.
41                 editableField(text: "Client Name", textFieldContent: $name, disabled: false)
42                 editableField(text: "Client Email", textFieldContent: $email, disabled: false)
43                 editableField(text: "Client Username", textFieldContent: $username, disabled: true)
44                 editableField(text: "Client Password", textFieldContent: $password, disabled: false)
45
46                 if isSaved {Section {Text("Saved.")}} //saved message
47
48                 Section(header: Text("Choose A Date To Send As JSON")){ //The datepicker to select the date
49                     DatePicker(selection: $dateToSendAsJson, displayedComponents: .date) {Text("")}.labelsHidden()
50                     Button("Send Records Of \((monthAsText)") { //A button to bring up the email sheet
51                         let handler = jsonHandler(months: self.months, monthToSend: self.components.month ?? 0, yearToSend:
52                             self.components.year ?? 0, client: self.client)
53                         handler.emailJsonData(showAlert: self.$alertShowing) //brings up the email sheet.
54                     }
55                 }
56                 .alert(isPresented: $alertShowing) { //alert appears if there are no records for the selected date. Thinking Ahead
57                     Alert(title: Text("There are no employees present this whole month. There is no information to be sent.))
58                 }
59                 .navigationBarItems(trailing: Button("Save Changes"){
60                     self.client.name = self.name
61                     self.client.email = self.email
62                     self.client.username = self.username
63                     self.client.password = self.password
64                     self.isSaved = true
65                     trySave(moc: self.moc)
66                 })
67                 .navigationBarTitle("Profile")
68                 .navigationBarBackButtonHidden(true)
69                 .navigationBarStyle(StackNavigationViewStyle())
70             }
71         }
72     }
```

## List of Logs (CRUD)

```
84     func saveLogChanges() {
85         //rewrite all logs and save the logs in the DB if there are changes
86         for index in 0...logs.count-1 {
87             day.logArray[index].date = logs[index]
88         }
89         if moc.hasChanges {
90             //if there are changes, save the new logs into the database and update the corresponding daily and monthly calculations
91             trySave(moc: moc)
92             logHandler(employee: employee, customDate: day.dayDate!, in: moc).refreshOvertimesAndWorktimes(for: day)
93             trySave(moc: moc)
94         }
95         saved = true //display saved message
96         added = false //dismiss added message if a log was added beforehand.
97     }
98
99     //deletes the log in the DB and the list.
100    func deleteLog(at offsets: IndexSet) {
101        for offset in offsets { //in case there are multiple offsets, use a for loop
102            //remove from the log from the DB, and refresh the calculations
103            logHandler(employee: employee, in: moc).deleteLog(log: day.logArray[offset])
104            logs.remove(at: offset) //remove from the list to update as well. Thinking Concurrently.
105        }
106        added = false
107        saved = false
108    }
109
110    func manualAddLog() {
111        //get the components of the chosen date in the datepicker.
112        let componentsToAdd = Calendar.current.dateComponents([.hour, .minute], from: dateToAdd)
113        //convert it into a Date data type
114        let actualDateToAdd = returnDateWithComponents(year: day.wrappedDate.year!, month: day.wrappedDate.month!, day: day.wrappedDate.day!,
115            hour: componentsToAdd.hour!, minute: componentsToAdd.minute!)
116        //add the log through the handler into the DB along with updating calculations.
117        let logger = logHandler(employee: employee, customDate: actualDateToAdd, in: moc)
118        logger.addNewLog()
119        logs.append(actualDateToAdd) //update on the list. Thinking Concurrently.
120        logs = logs.sorted() //sort the dates in ascending order of dates when added.
121        added = true //display added message
122        saved = false //dismiss saved message if changes made beforehand.
123    }
```

## List of Employees (CRUD)



The image shows a mobile application interface for adding a new employee. At the top, there is a status bar with the time 1:22 and signal indicators. Below this, a blue header bar contains the text 'Add Employee' in white. The main title 'Add New Employee' is displayed in a large, bold, black font. The form consists of several input fields: 'Name:' with a placeholder 'Enter Name Of New Employee Here', 'Username:' with a placeholder 'Enter Username Here', and 'Password:' with a placeholder 'Enter Password Here'. Below these is a 'Type' section with a list of employee types: 'Normal Pay Employee', 'No Extra Pay Employee', 'Diligence Pay Employee', and 'Fixed Extra Pay Employee'. The 'Normal Pay Employee' option is selected and highlighted with a blue background. At the bottom of the form, there is a red error message that reads 'Please enter a name.'.

1:22

Add Employee

### Add New Employee

Name:

Enter Name Of New Employee Here

Username:

Enter Username Here

Password:

Enter Password Here

Type

- Normal Pay Employee
- No Extra Pay Employee
- Diligence Pay Employee
- Fixed Extra Pay Employee

Please enter a name.

*Adding Sheet*

```

54     func createNewEmployee(context: NSManagedObjectContext) {
55         let newEmployee = Employee(context: context)
56         newEmployee.name = name
57         newEmployee.username = username
58         newEmployee.password = password
59         newEmployee.type = type
60     }
61
62     func handleAddingEmployee(presentation: Binding<PresentationMode>, moc: NSManagedObjectContext) -> String
        {
63         if checkAddingError() == .none {
64             createNewEmployee(context: moc)
65             presentation.wrappedValue.dismiss() //for the UI
66             trySave(moc: moc)
67         }
68         return checkAddingError().rawValue
69     }
70 }

```

### Creating Employees

Table: ZEMPLOYEE							
	Z_PK	Z_ENT	Z_OPT	ZNAME <sup>1</sup>	ZPASSWORD	ZTYPE	ZUSERNAME
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	433	3	2	Amanda A	213	Normal Pay Employee	Amanda
2	419	3	4	Amin A	123	Normal Pay Employee	Amin
3	422	3	4	Bill Bones	123	Normal Pay Employee	Bill

The Employee Database table.