University Dental Clinic System

3rd Deliverable

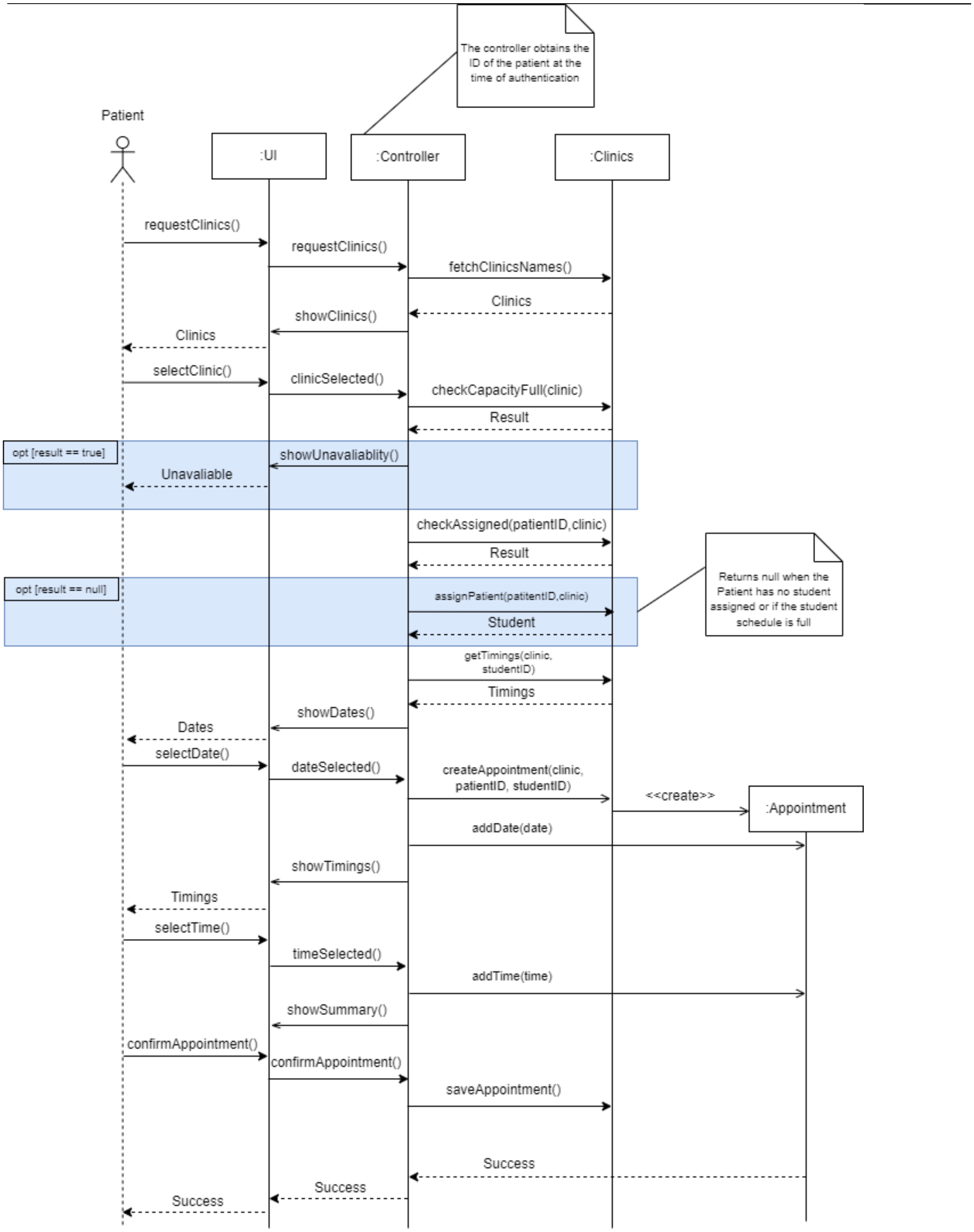
COE 420 Software Engineering
American University of Sharjah
College of Engineering
Computer Science and Engineering

Group 4
Joseph Press
Gregory Smith
Ayah Almheiri
Abdu Sallouh

Contents

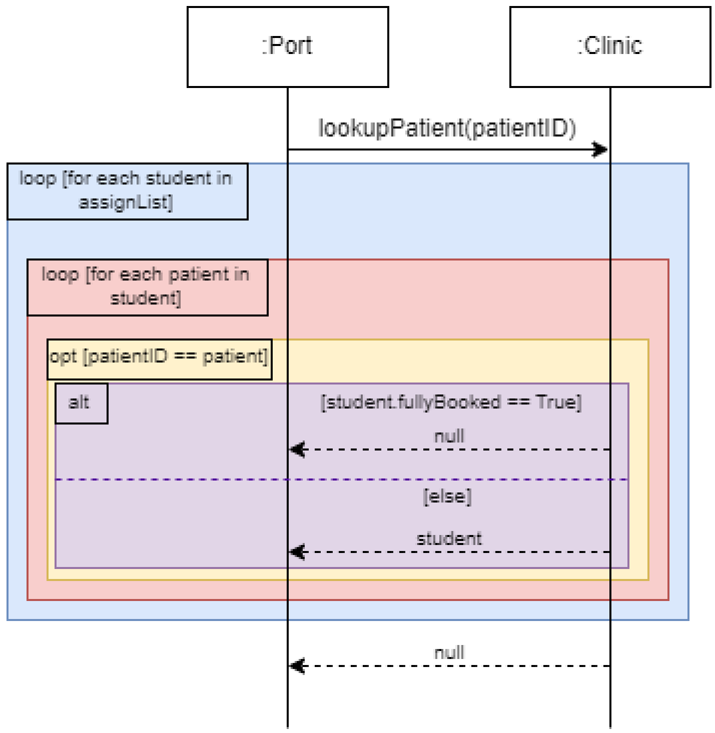
1. Sequence Diagrams

1.1. Main Flow

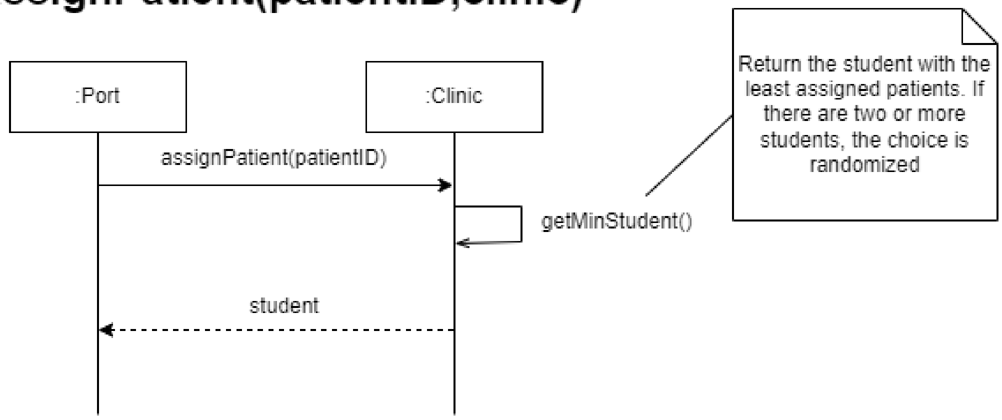


1.2. Clinic Subsystem Sequences

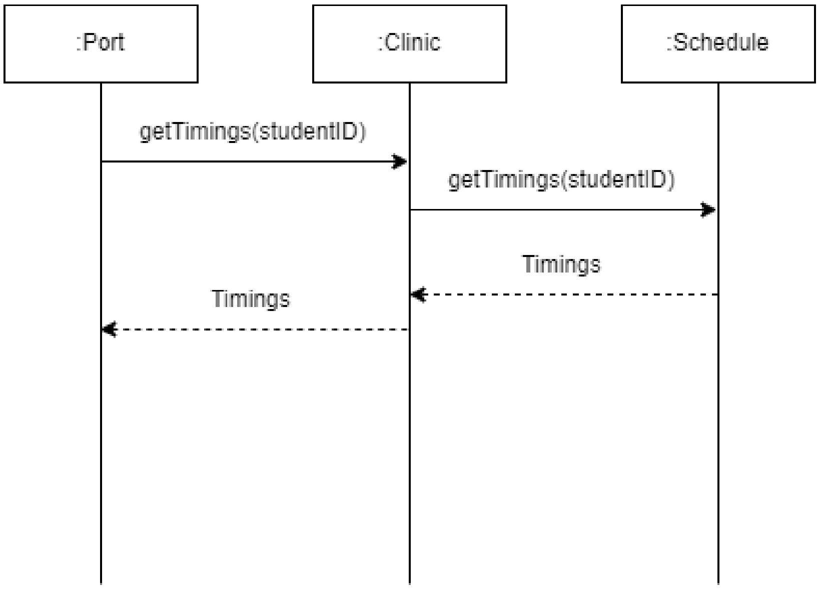
checkAssigned(patientID, Clinic)


:Port
:Clinic
lookupPatient(patientID)
loop [for each student in assignList]
loop [for each patient in student]
opt [patientID == patient]
alt
[student.fullyBooked == True]
null
[else]
student
null

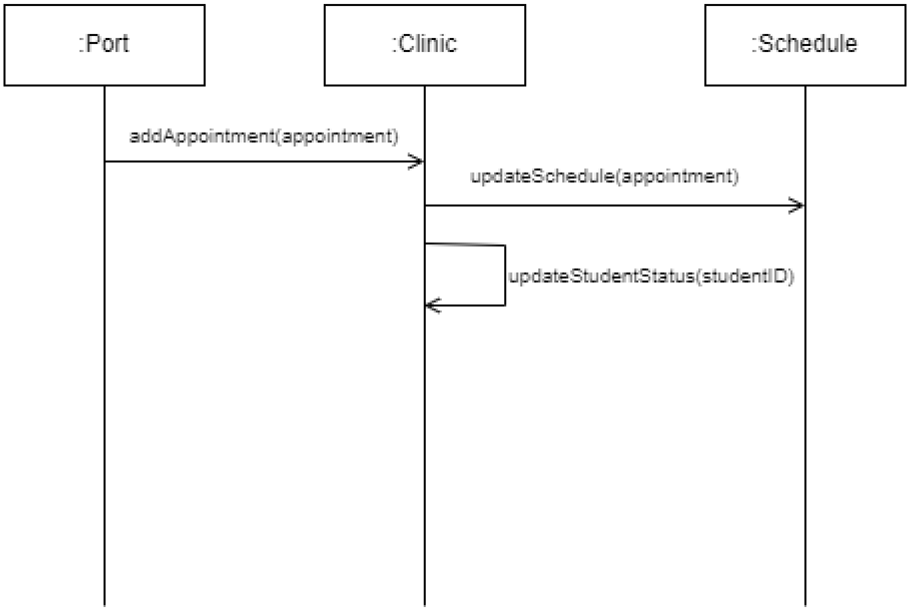
assignPatient(patientID,clinic)


:Port
:Clinic
assignPatient(patientID)
getMinStudent()
student
Return the student with the least assigned patients. If there are two or more students, the choice is randomized

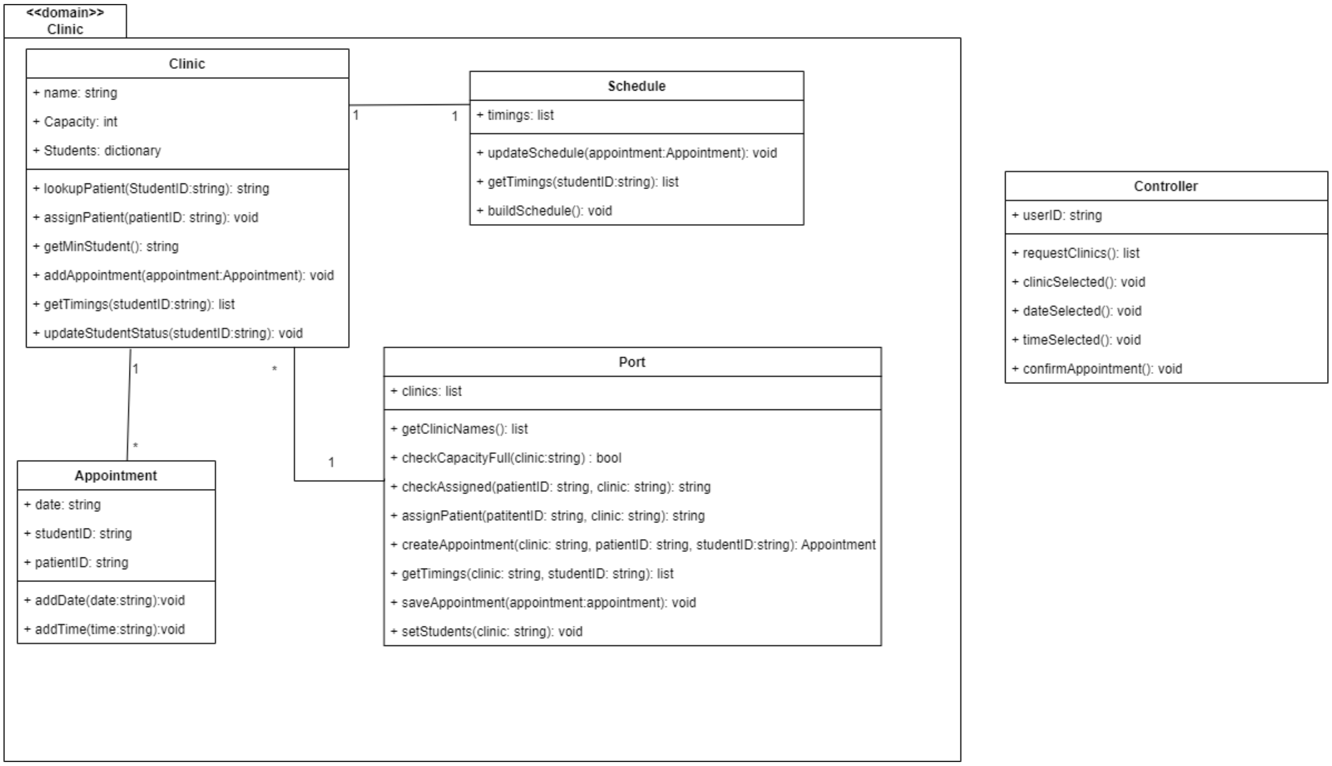
getTimings(clinic,studentID)



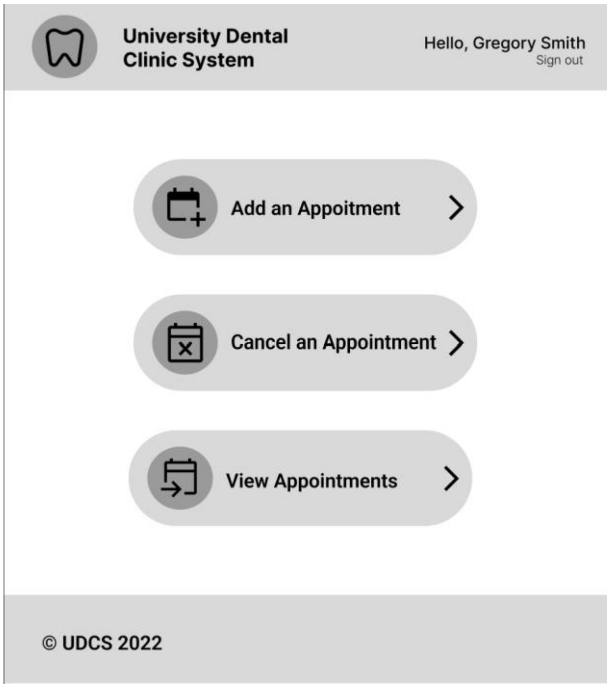
saveAppointment()

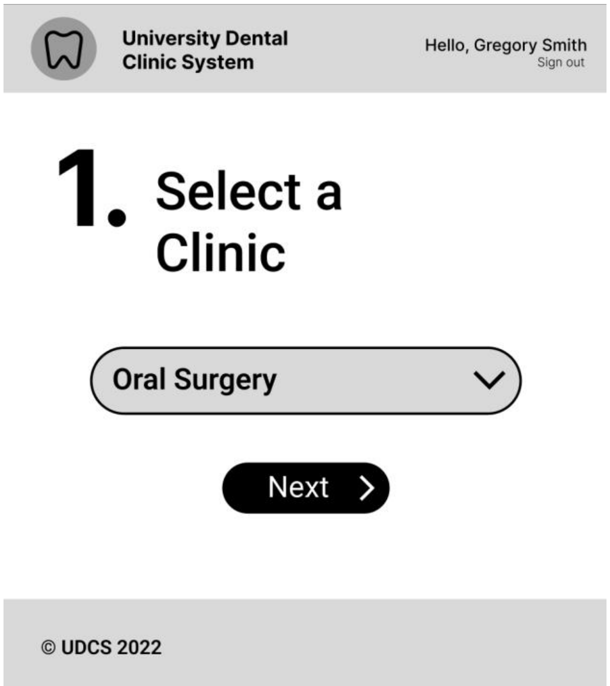

:Port
:Clinic
:Schedule
addAppointment(appointment)
updateSchedule(appointment)
updateStudentStatus(studentID)

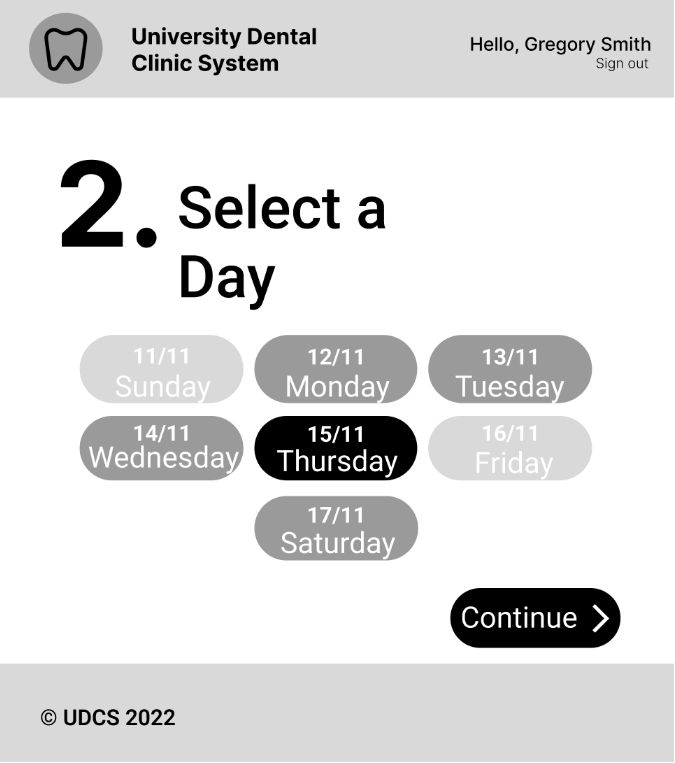
2. Detailed-Class Diagram


<<domain>>
Clinic
Clinic
+ name: string
+ Capacity: int
+ Students: dictionary
+ lookupPatient(StudentID:string): string
+ assignPatient(patientID: string): void
+ getMinStudent(): string
+ addAppointment(appointment:Appointment): void
+ getTimings(studentID:string): list
+ updateStudentStatus(studentID:string): void
1
1
Schedule
+ timings: list
+ updateSchedule(appointment:Appointment): void
+ getTimings(studentID:string): list
+ buildSchedule(): void
Controller
+ userID: string
+ requestClinics(): list
+ clinicSelected(): void
+ dateSelected(): void
+ timeSelected(): void
+ confirmAppointment(): void
1
*
*
1
Port
+ clinics: list
+ getClinicNames(): list
+ checkCapacityFull(clinic:string) : bool
+ checkAssigned(patientID: string, clinic: string): string
+ assignPatient(patitentID: string, clinic: string): string
+ createAppointment(clinic: string, patientID: string, studentID:string): Appointment
+ getTimings(clinic: string, studentID: string): list
+ saveAppointment(appointment:appointment): void
+ setStudents(clinic: string): void
Appointment
+ date: string
+ studentID: string
+ patientID: string
+ addDate(date:string):void
+ addTime(time:string):void

3. GUI Snapshots





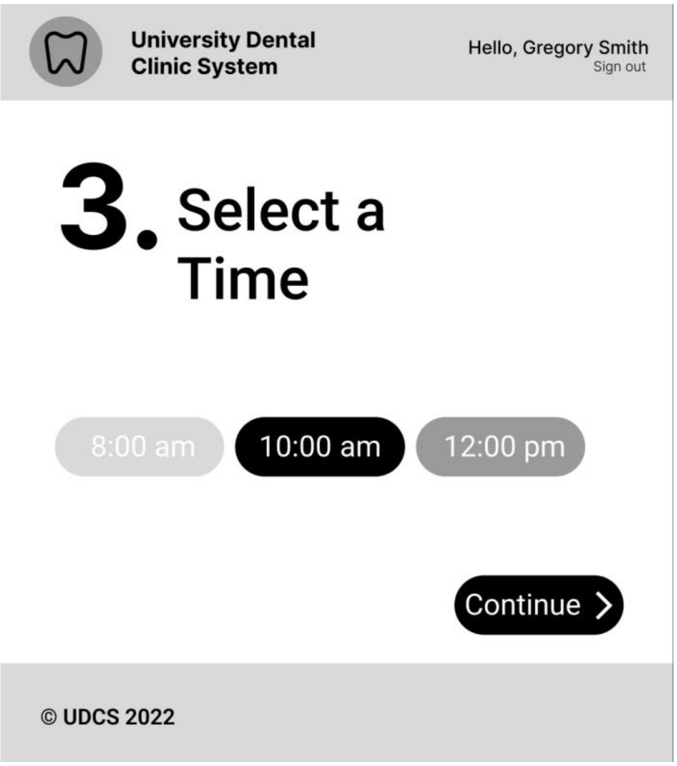


University Dental Clinic System

Hello, Gregory Smith
Sign out

2. Select a Day

© UDCS 2022

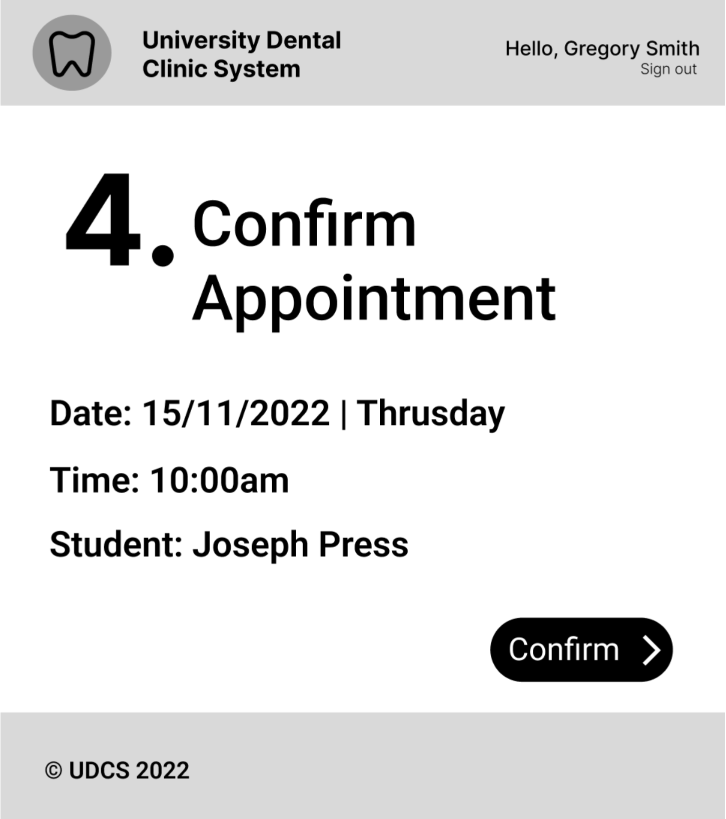


University Dental Clinic System

Hello, Gregory Smith
Sign out

3. Select a Time

© UDCS 2022

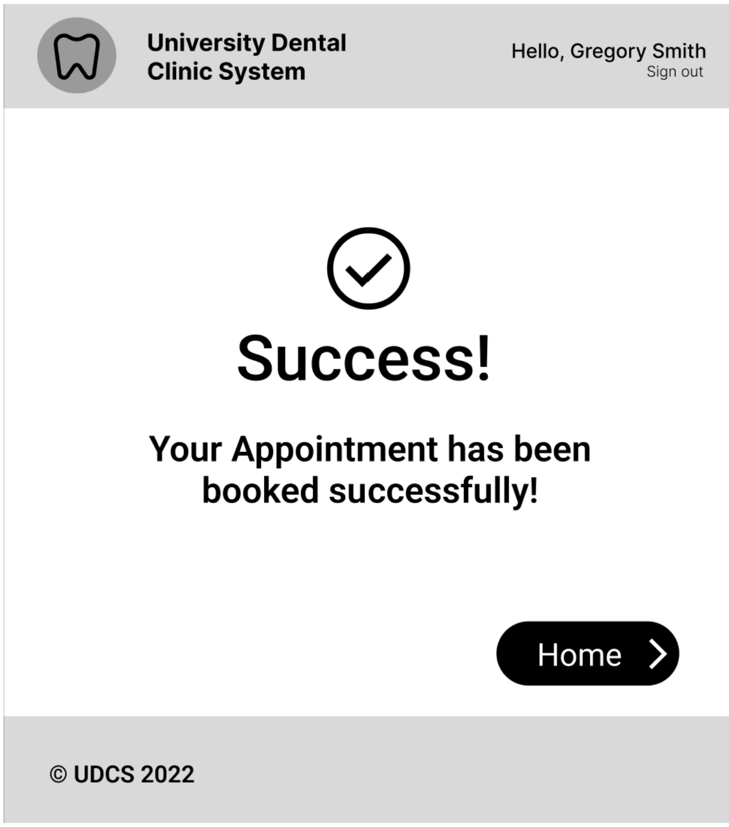


4. Confirm Appointment

Date: 15/11/2022 | Thrusday

Time: 10:00am

Student: Joseph Press



Success!

Your Appointment has been booked successfully!

4. Testing

4.1. Blackbox Testing

a. State Transition Testing

- In terms of black box testing the tester would test the use-case scenario while certain scenarios are being set such as:

1. **Booking an appointment at a clinic for the first time**
2. **Booking an appointment at a clinic recurrently (2nd Booking or more)**
3. **Booking an appointment at a clinic that has a filled capacity**
4. **Booking an appointment at a clinic when the assigned student has a full schedule for the week.**

- These scenarios are chosen on the basis of trying to cover the largest amount of alternative pathways that a patient may undertake when booking an appointment at a certain clinic.
- The tester would observe the behaviour of the state, the transition of states, and the overall performance of each state in terms of its responsiveness.
- The Black-box testing would also consider the students' side of view whereby tests would be conduced from tester using the student account privilege to check whether their schedules were updated and any new appointments were added to them. Tester may test the following Scenarios:

5. **Students viewing the clinic schedule when its empty.**
6. **Students viewing the clinic schedule after a patient made an appointment.**
7. **Students viewing the clinic schedule when it's full.**
8. **Students viewing the patients they have been assigned to in the clinic.**

b. BVA Testing

- Given that the dates and times that are presented are always valid timings boundary testing may incur on only the first valid and the valid dates and timings when booking an appointment. So, boundary testing may include:

1. **(First Valid Date, First Valid Time) - Valid**
2. **(First Valid Date, Last Valid Time) - Valid**
3. **(Last Valid Date, First Valid Time) - Valid**
4. **(Last Valid Date, Last Valid Time) - Vallid**

4.2. Whitebox Testing

- White box testing can be used to test whether the methods used in component match the desired outcome through which coverage testing is used to test all lines of code in the system whereby test-cases would be used. Examples of test cases would be:
  - **Asserting the output of methods involved in the use-case**
  - **fetchClinicsNames() : list output would be tested by comparing the provided list with the actual list**
  - **checkCapacityFull(clinic) : bool output would be tested whereby the clinic capacity is modified and expected output is compared with actual output**
  - **createAppointment(clinic, patientID, student) : appointment would be tested whereby the output of the method would be tested against an expected output.**
  - **Methods involve the patient input such as selectDate(), selectTime(), and selectClinic() would be tested whereby the selected input would be validated and tested against expected output().**
  - **Methods that are invoked in the UI such as showDates(), showTimings() etc.. would be testing whereby the actual output is tested against expected output on the screen.**

4.3. Automation

- Automation system can be used to simulate a real working of the use-case where test-agents book appointments at random times at different clinics with a rule base that determine expected results which are then compared to actual results found through automation. Automation tool such as **pytest** can be used to carry testing automation.