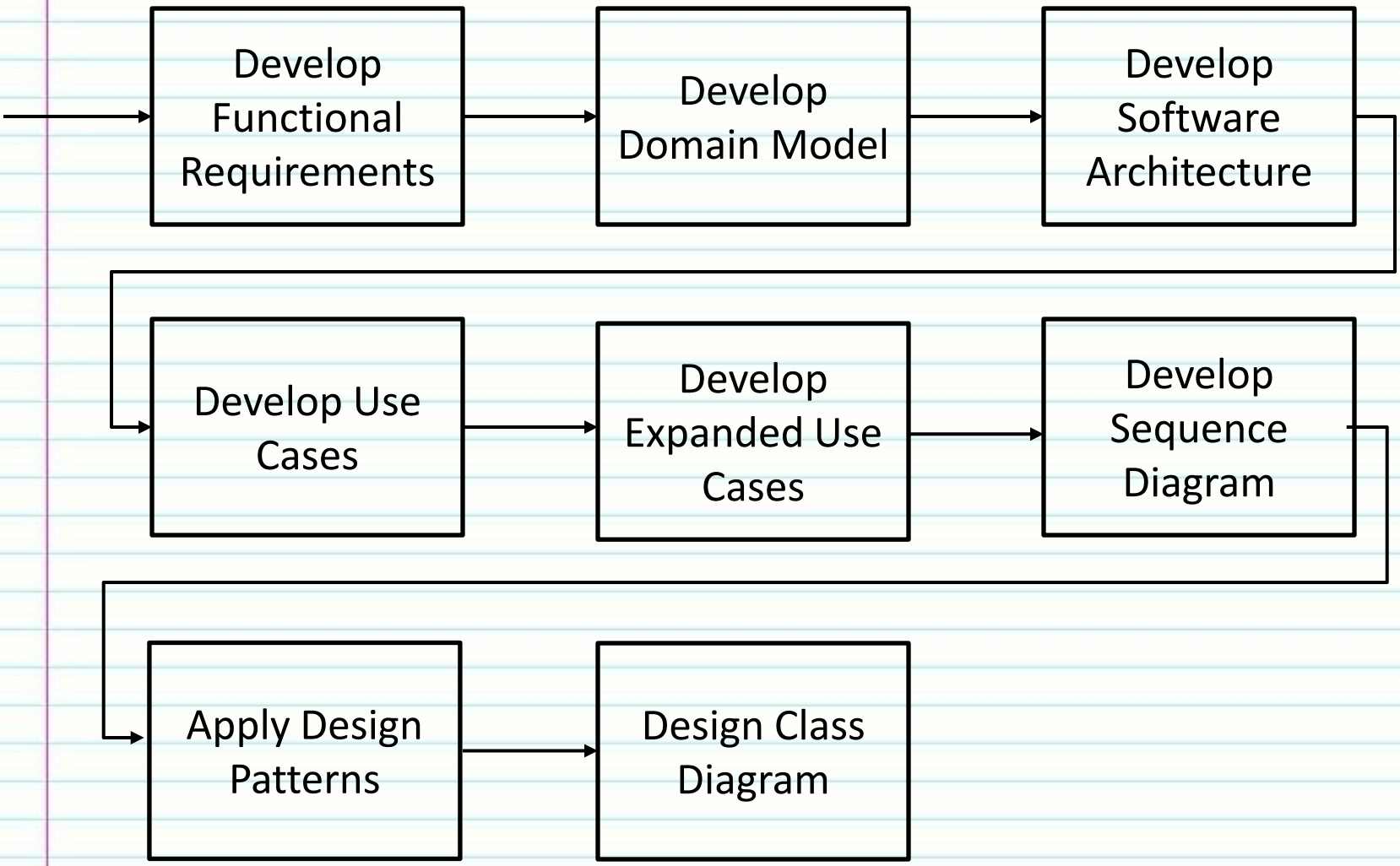
Chapter 8 – Actor-System Interaction Modeling

Dr John H Robb, PMP, IEEE SEMC
UT Arlington
Computer Science and Engineering

Key Takeaway Points

- Actor-system interaction modeling is modeling and design of how the system interacts with the actors to carry out the use cases.
- Actor-system interaction modeling is accomplished by constructing a two-column table that describes, for each interaction, the actor input and actor action, and the system response.

Book Approach to OOSE



Use Case Modeling Steps

planning phase

iterative phase

requirements

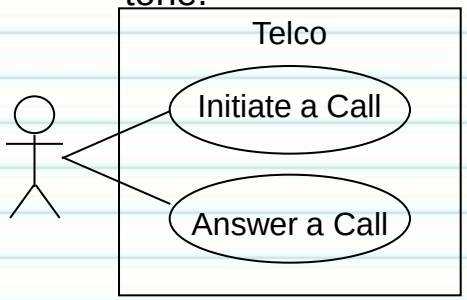
Deriving use cases from requirements

abstract use cases (e.g., Initiate a Call)

Defining use case scope

abstract & high level use cases

Example high level use case:
TUCBW caller picks handset from base
TUCEW caller hears the ring tone.



Depicting use case contexts

We've already done these steps

We're ready to do this

Actor: Caller	System: Telco
1. TUCBW caller picks up the handset.	2. The system generates a dial tone.
3. The caller dials each digit of the phone number.	4. The system responds with a DTMF tone for each digit dialed.
5. The caller finishes dialing.	6. The system produces the ring tone.
7. TUCEW the caller hears the ring tone.	

Specifying actor-system interaction (expanded use cases)

Review: Three Levels of Use Case Specification

1) Abstract use case: using *a verb and a noun* phrase

2) High level use case: stating exactly *when and where* the use case begins and *when* it ends using

   TUCBW ... (This use case begins with ...)

   TUCEW ... (This use case ends with ...)

3) Expanded use case: describing *step by step how the actor and the system interact* to accomplish the business task using a *two column table* - *this is a step by step choreograph between the actor and the system*

   a. *Always starts with the System displaying some information (status/message/web page) as Step 0*

   b. *Step 1 picks up the TUCBW*

   c. *Always ends on an actor step with confirmation that the actor sees/acknowledges the use case has provided the desired response. This is the TUCEW*

   d. *It always has a pre-condition and post-condition (discussed next)*

Design By Contract

Design by Contract

- Is a software correctness methodology (formal proof of correctness) which prescribes that software designers should define formal, precise and verifiable interface specifications for software components
- These specifications are referred to as "contracts", in accordance with the conditions and obligations of business contracts.
- There are a number of programming languages that implement the pre and post-conditions that provides these verifiable specifications (Eiffel, extensions to Java, extensions to C++)
- In this course (SE1) we concentrate on using these as a design notation where the power in these notations is in the creation of very crisp and precise interface specifications during design

The contract to produce quality software

- Use cases provide a language for describing requirements that aim to be understandable to both technical (developers) and non-technical (customers and users) people.
- A significant part of a use case consists of the pre-condition and post-condition that constrain it. Pre-conditions, post-conditions are collectively known as **assertions**.
- We use these in the Expanded Use Cases (Exp UCs) to create a much more crisp and precise interface specification for each Exp UC
- TUCBW/TUCEW focus specifically on Actors - Pre/post conditions focus on the System (software) what it requires and what it will provide.
  - Together, these provide a much more specific description of the interface of an Exp UC

The contract to produce quality software (cont.)

- Useful Pre-conditions in the Expanded Use Case
  - They tell us about assumptions we are making - for example, that the user has registered before logging in.
  - Another example - that the user has logged into the system - this tells us that they are recognized as a valid user and have passed all verification checks.
  - Last example - that the system requires the actor to have selected a restaurant
  - The pre-conditions allow us to create a very specific separation of concerns - things that we are not going to address in the Exp UC

The contract to produce quality software (cont.)

- Useful Post-conditions in the Expanded Use Case
  - They tell us what functions or results the system guarantees to occur when the Exp UC has met all of its pre-conditions.
  - For example, a search UC might have a post-condition that it will provide a sorted list of apartments or a message indicating that none exists (null list).
  - Another example, the UC for Login guarantees that the user will have verified credentials and is able to use the system (we're doing Sunny Day scenarios in this class)

The contract to produce quality software (cont.)

- TUCBWs/TUCEWs together can show us global scoping issues
  - We can use the TUCEW from one Exp UC and the TUCBW from another to show scope issues
  - This occurs where the Actor actions and results accidentally overlap or have gaps between ExpUCs

- Pre/post-conditions show us where system (software) global scoping issues might exist
  - We can use the post-condition from one Exp UC and the pre-condition from another to show functional issues
  - This occurs where the post-condition and pre-conditions accidentally overlap or have gaps between ExpUCs

Expanded Use Case with Pre/Post-Conditions

***Precondition**: The system shows the user logged in and it is showing the user the main page.*	
Actor: Staff User	System: SAMS
	0. System displays the staff main page.
1. **TUCBW** the staff user selects the “Add Program” function.	2. System displays the Add Program page.
3. The staff user enters program detail and clicks the ”submit" button.	4. System checks the submitted info and shows a confirmation message if no error is found.
5. **TUCEW** the staff user confirms the message	
***Postcondition**: The system adds the new program and it is immediately available for search.*	

Try to start pre and post conditions with the phrase “The system…”

Expanded Use Case with Pre/Post-Conditions

***Precondition**: The system shows the user logged in and it is showing the user the main page.*	
Actor: Staff User	System: SAMS
	0. System displays the staff main page.
1. **TUCBW** the staff user selects the “Add Program” function.	2. System displays the Add Program page.
3. The staff user enters program detail and clicks the ”submit" button.	4. System checks the submitted info and shows a confirmation message if no error is found.
5. **TUCEW** the staff user confirms the message	
***Postcondition**: The system adds the new program and it is immediately available for search.*	

In the HL UCs we tried to avoid mentioning buttons and clicks, but after developing the UI prototypes we use them in the **non-TUCBW/TUCEW steps**

Typical Student Errors with Pre/Post Conditions

- Pre or Post condition - "None"
  - Pre-condition: Are you saying that a non-logged in user can use the system?
  - Post-condition: Are you really saying the software does nothing?
- Duplication of TUCBW/TUCEW
  - There are different perspectives here - the TUCBW/TUCEW is what the Actor does and sees
  - The pre/post conditions are what the system expects and provides
- Imposing invalid constraints in the pre-condition
  - Constraints on the actors "The user of this Exp UC shall have a positive balance in their student account." - pre/post conditions are on the system not users

➢ Use them to state the definitive thing the Exp UC guarantees to provide from a system standpoint

➢ Use the Use Case Interaction diagram to expose flow issues related to pre/post conditions of system processing

User Interface Prototypes

Showing User Interface Prototypes

- In the requirements we focused on creating a short functional statement of using attributes, doing something with them, and producing attributes

- Each Exp UC will have **UIPs for each System side step**
  1. To show the attributes used in performing the function of the EUC (this is where the user sees the data that needs to be entered)
  2. To show what attributes are produced (this is where the user sees the data that is produced)

- These related directly back to the attribute tables captured in the requirements

- This section of the course takes those tables and provides the user with their first view of what the screens look like

Showing UI Prototypes w/ Expanded Use Case

UI Prototype always shown **only** on the System Side!

UC 1: Signup

Precondition: The system does not have a registered account in the UTA Shared Ride System for the user.	
Actor: Commuter/Rider	**System: UTA Shared Ride**
	0. System displays the Login page.
1. **TUCBW** the Commuter/rider will be able to select the *Create a new account* link on Login Page.	2. System display Signup form in Signup page. (Refer Figure 1)
3. Commuter/Rider fills the form and clicks on *Sign Me Up* button.	4. System displays Signup Successful message and Commuter/Rider is redirected back to Login Page.(Refer Figure 2, Figure 3)
5. TUCEW the Commuter/Rider gets the access and can view their function list	
Post condition: The system creates a new account for Commuter/Rider.	

Showing UI Prototypes w/ Expanded Use Case (cont.)

Figure 1:

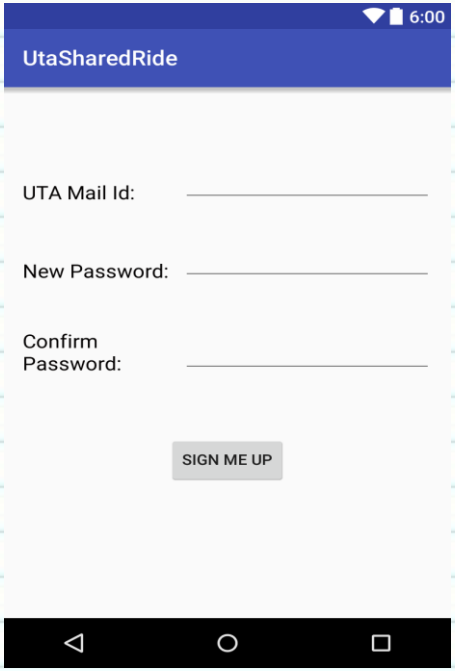


Figure 2:

Showing UI Prototypes w/ Expanded Use Case (cont.)

Figure 3:

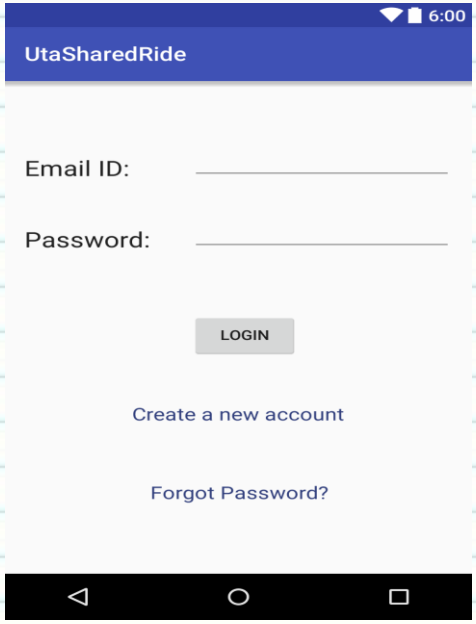
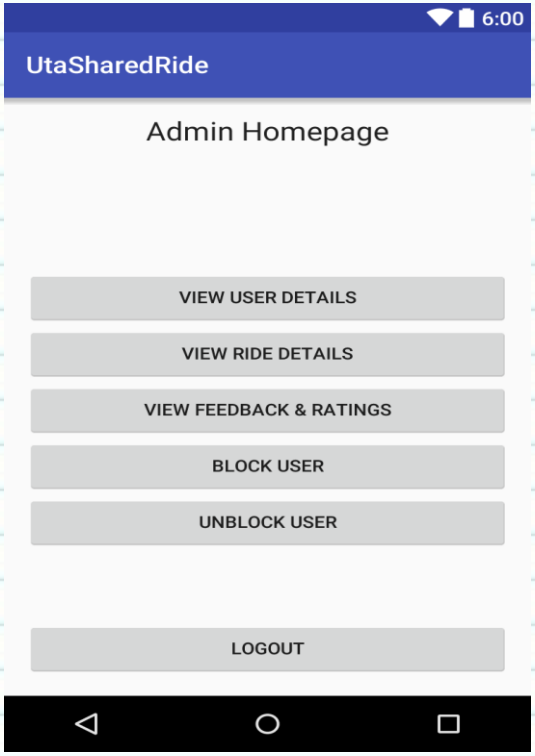


Figure 4:



Guidelines for Expanded Use Case

- Expanded use cases inherit all the guidelines for high level use cases (because expanded use cases are refinements of high level use cases).
- Expanded use case specification should state a user interface that a novice actor can easily begin
  - example: “0) System displays the homepage.”
- Expanded use cases may show prototypes of the user interfaces
  - for soliciting feedback from the customer/user
  - to inform the programmer what the UI should look like
- Expanded use case specification should always end with the actor, preferably with an actor action to acknowledge that the system has accomplished the task properly.
- Expanded use case specification must include pre- and post-conditions to explicitly specify the assumptions and effect of the use case.

Preparing for Design Patterns

- In the next Module (M08) we will study sequence diagrams, but in order to avoid some re-work we want to start thinking about these terms in the expanded use cases before we apply design patterns to them (M09)
- These terms will help us make the jump to the design patterns much more easily
- Start to think of each Expanded Use Case utilizing the following three components (useful when we get to Sequence Diagrams)
  - A GUI (e.g. Add Program GUI)
  - A Controller (e.g., Add Program Controller)
  - A Database Manager (e.g., DBMgr)
- Each Exp UC will have its own GUI, Controller, and DBMgr which later can be combined with other Exp UCs (if necessary)
- When we develop sequence diagrams we will use these components for each Exp UC - M09 Design Patterns will explain why these are important

Expanded Use Case Summary

- We have learned three important principles
  1. How to develop an Expanded Use Case
     a. Always starts with the System displaying some information (status/message/web page) as Step 0
     b. Step 1 picks up the TUCBW
     c. Always ends on an actor step with confirmation that the actor sees/acknowledges the use case has provided the desired response. This is the TUCEW
     d. It always has a pre-condition and post-condition (discussed next)
  2. Each Expanded Use Case have two User Interface Prototypes
     a. They are always shown only on the System side but they represent the input the user provides and the output he sees
     b. They help use the attribute tables from the requirements
  3. Each Expanded UC will have in the Sequence Diagrams the following three components
     a. UC GUI, UC Controller, DB Mgr

UI Prototypes and the Project

- Project teams need to develop the UI Prototypes in Iteration 2.1 for UCs targeted to Iteration 1 and 2
  - Iteration 2 is a lot of work with UI Prototypes and Sequence Diagrams - the smartest thing to do is the following
    1. **Start developing the XML for the UI prototypes for Iterations 1 and 2 now - divide this up among your teammates immediately and get started**
    2. Don't worry about the functional code yet - just get the XML to match the attributes in the requirements and Domain model
    3. Stick to the attributes in the requirements and Domain model - don't change these unless you absolutely have to - you want to avoid iterating as much as possible!
- You can change your Increment Matrix at any time to better distribute the workload between Iterations
  - BUT anything you push off now is still due later

Example Student Project

UC 1: Signup

Precondition: The system does not have a registered account in the UTA Shared Ride System for the user.	
Actor: Commuter/Rider	**System: UTA Shared Ride**
	0. System displays the Login page.
1. **TUCBW** the Commuter/rider will be able to select the *Create a new account* link on Login Page.	2. System display Signup form in Signup page. (Refer Figure 1)
3. Commuter/Rider fills the form and clicks on *Sign Me Up* button.	4. System displays Signup Successful message and Commuter/Rider is redirected back to Login Page.(Refer Figure 2, Figure 3)
5. TUCEW the Commuter/Rider gets the access and can view their function list	
Post condition: The system creates a new account for Commuter/Rider.	

Example Student Project (cont.)

UC 2: Login

Precondition: The system has an account in the UTA Shared Ride System for the user	
Actor: User	**System: UTA Shared Ride**
	0. System displays the Login page (Refer Figure 3).
1. **TUCBW** the user enters the UTA email id, password and selects "Login" in their function list.	2. The system displays different pages in different scenarios. (I) Admin will be shown Admin home page. (Refer Figure 4) (ii) First time Commuter/Rider will be shown Create Profile page. (Refer Figure 5) (iii) Regular Commuter/Rider will be shown Homepage. (Refer Figure 6)
3**. TUCEW** the user gains access into the system. System Users see their function list	
Post condition: The system creates a user account where he is eligible to access the appropriate user functions.	

Example Student Project (cont.)

UC 2.1: Reset Password

Precondition: The system has an account with a completed profile in the UTA Shared Ride System for the user. The system is showing the application main function	
Actor: Commuter/Rider	**System: UTA Shared Ride**
	0. System displays the Login page.
1. **TUCBW** selects "Forgot Password?" link on his function list.	2. System displays Reset Password page. (Refer Figure 7)
3. Commuter/Rider enters his/her registered UTA Email ID and clicks on Reset Password button.	4. System generates a random password; email it to the registered UTA email ID. System also displays the message mentioning that new password has been emailed.(Refer Figure 8)
5**. TUCEW** the Commuter/Rider sees the new password has been emailed message.	
Post condition: The system updates the password of Commuter/Rider with the random password which is mailed to the Commuter/Rider.	

Example Student Project (cont.)

UC 3.1: Create User Profile

Precondition: The system has an account for the user but the profile information is not yet created. The system shows the Commuter/Rider as logged in.	
Actor: Commuter/Rider	**System: UTA Shared Ride**
	0. System displays the Create Profile page. (Refer Figure 5)
1. **TUCBW** the new Commuter/Rider fills in 'create profile' form and submits the information.	2. System creates a new profile for the Commuter/Rider with the given values. System also displays the message saying profile creation is successful. (Refer Figure 9)
3**. TUCEW** the Commuter/Rider sees the successfully created profile message.	
Post condition: The system creates a new profile for Commuter/Rider where the Commuter/Rider can now access full functionalities of the system.	

Example Student Project (cont.)

UC 3.2: Update User Profile

Precondition: The system shows the Commuter/Rider as logged in.	
Actor: Commuter/Rider	**System: UTA Shared Ride**
	0. System displays the Home page.
1. **TUCBW** the Commuter/Rider clicks on the Profile Management Menu on the Home Page.	2. System shows the Update Profile submenu. (Refer Figure 10)
3. Commuter/Rider clicks on Update Profile Submenu.	4. System displays the update profile form. (Refer Figure 11)
5. The Commuter/Rider can update one or more fields and clicks on Update button.	6. System updated the user profile with new values; a message will be displayed to the Commuter/Rider saying update is successful.(Refer Figure 12)
7**. TUCEW** the Commuter/Rider sees the successfully updated profile message.	
Post condition: The system will update the values of the Commuter/Rider	

Example Student Project (cont.)

UC 4: Admin View

Precondition: The system shows the Admin as logged in.	
Actor: Admin	**System: UTA Shared Ride**
	0. System displays admin homepage.
1. **TUCBW** the admin selects the admin view function from his function list	2. System shows the view user details menu, view ride details menu, view feedback and ratings menu, block user and unblock user. (Refer Figure 4)
3. **TUCEW** the admin sees the admin view functions	
Post condition: The system displays admin functions on the admin view list	

Example Student Project (cont.)

UC 4.1: View User Details

Precondition: The system shows the Admin as logged in.	
Actor: Admin	**System: UTA Shared Ride**
	0. System displays admin homepage.
1. **TUCBW** the admin selects the view user details menu.	2. System shows the commuter details and rider details submenus.
3. Admin clicks on rider details or commuter details submenu.	4. System displays details based on the click a) Commuter details (Refer Figure 43) b) Rider details (Refer Figure 44)
5. Admin selects single record for commuter or rider.	6. System displays details of selected record. (Refer Figure 45. Refer Figure 46
7. **TUCEW** the admin sees the details of commuter or rider.	
Post condition: The system provides all the details of the selected commuter /rider.	

Example Student Project (cont.)

UC 4.2: View Ride Details

Precondition: The system shows the Admin as logged in.	
Actor: Admin	**System: UTA Shared Ride**
	0. System displays admin homepage.
1. **TUCBW** the admin will be able to click on the view ride details menu.	2. System shows the ride details. (Refer Figure 41)
3. Admin selects single record for ride details.	4. System displays details of selected ride. (Refer Figure 42)
5. **TUCEW** the admin sees the ride details.	
Post condition: The system provides all the details of the selected ride.	

Example Student Project (cont.)

Figure 1:

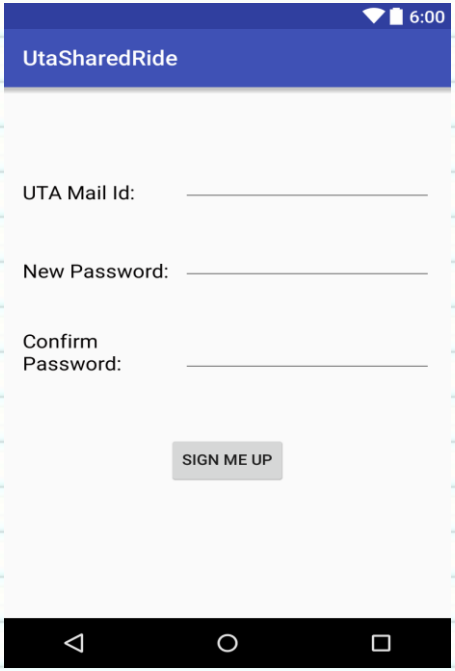


Figure 2:

Example Student Project (cont.)

Figure 3:

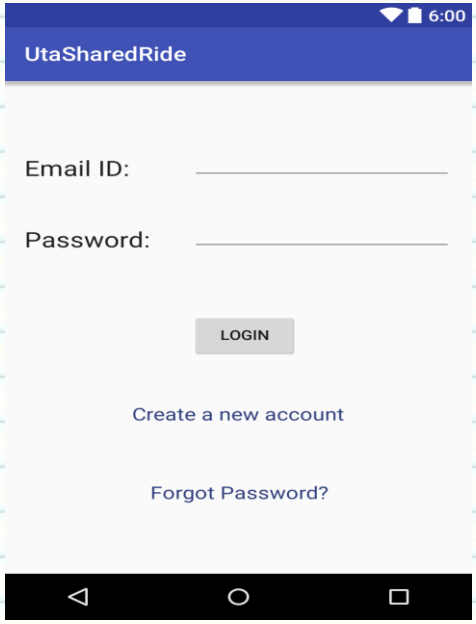
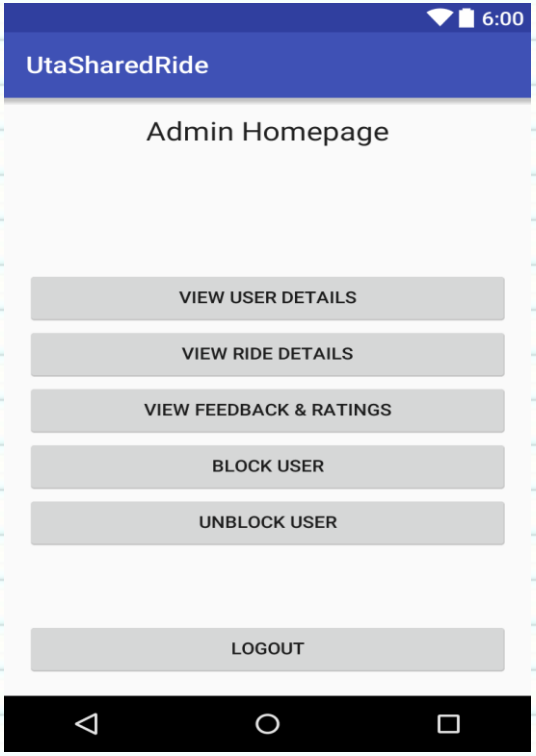
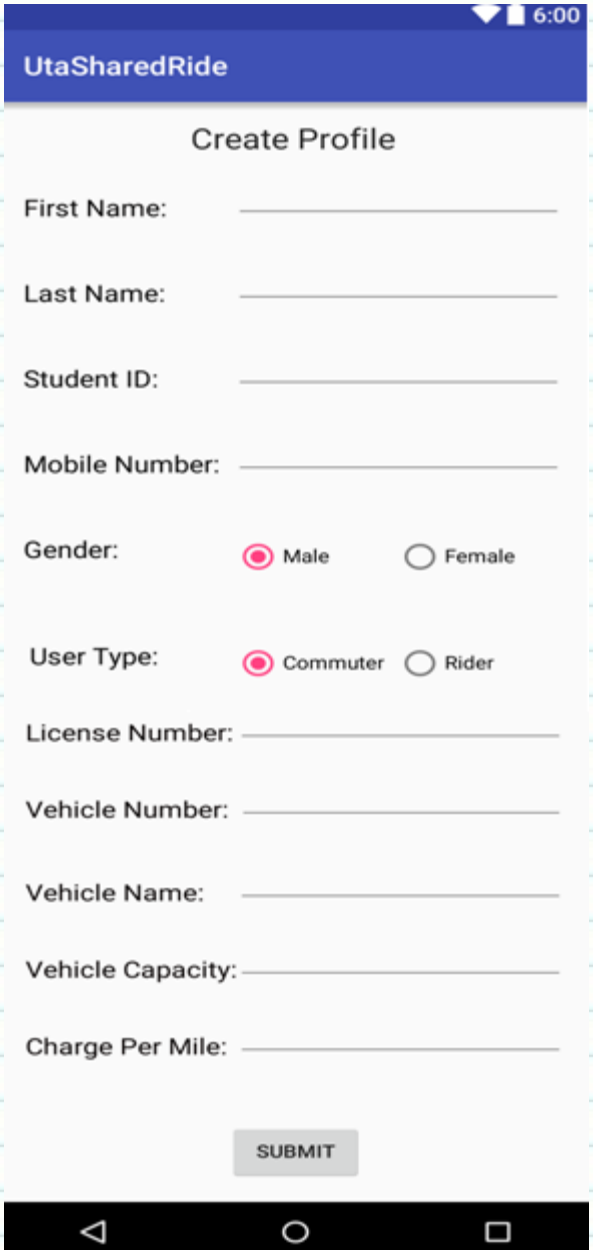


Figure 4:



Example Student Project (cont.)

Figure 5:



Example Student Project (cont.)

Figure 6:

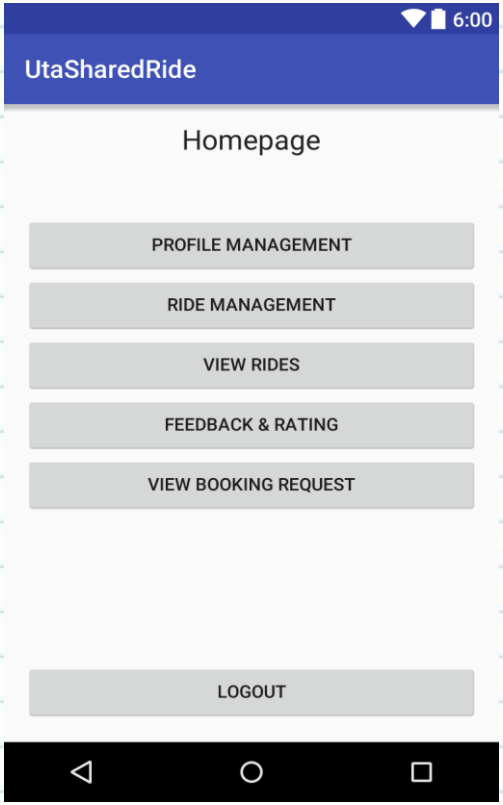


Figure 7:

Example Student Project (cont.)

Figure 8:

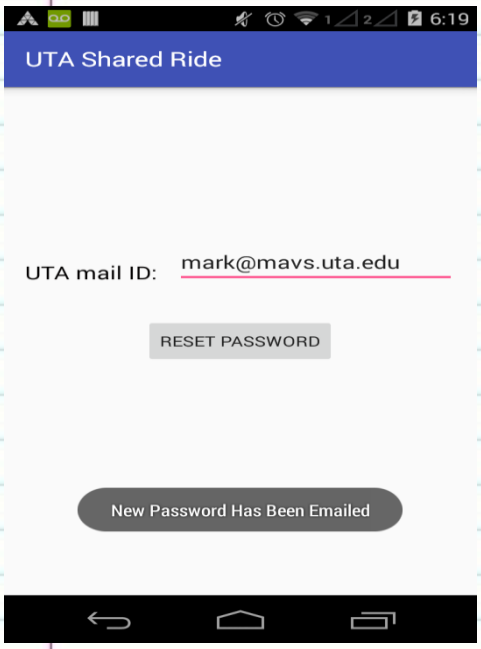
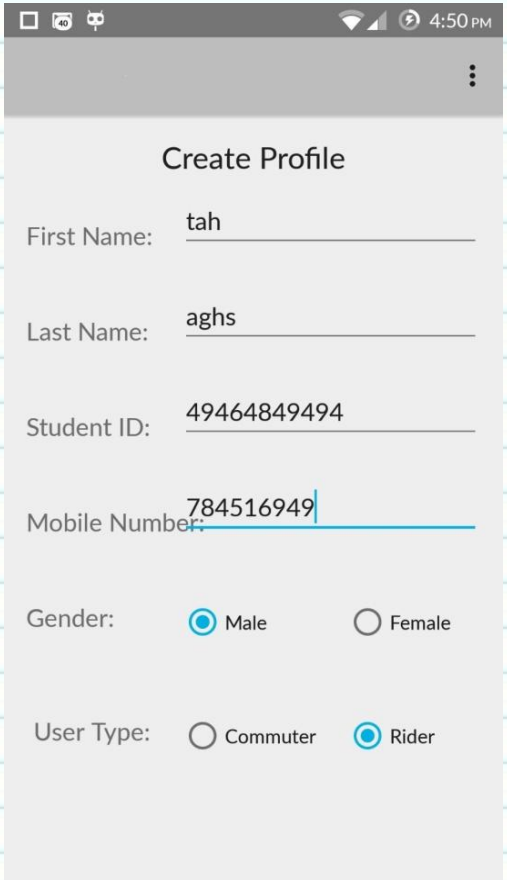
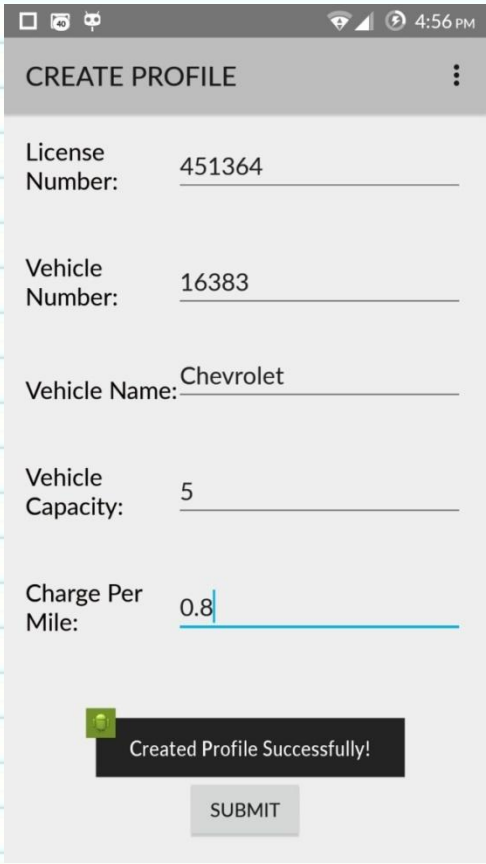


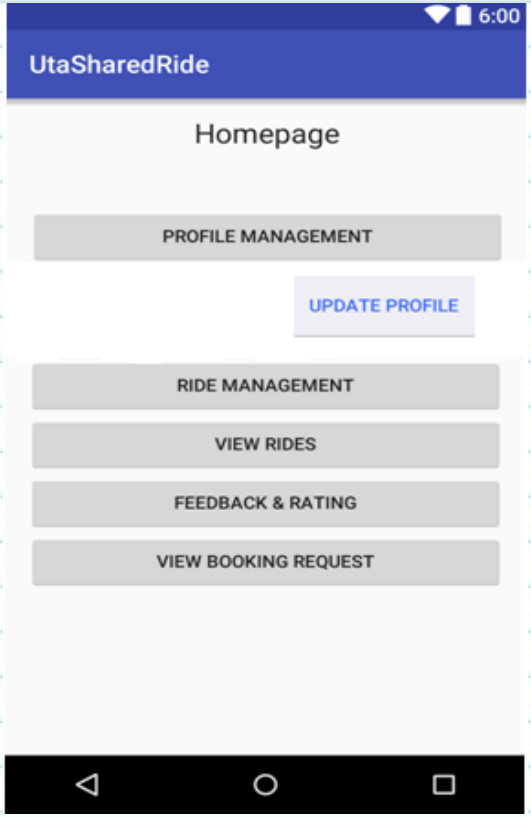
Figure 9:





Example Student Project (cont.)

Figure 10:



Example Student Project (cont.)

Figure 11:

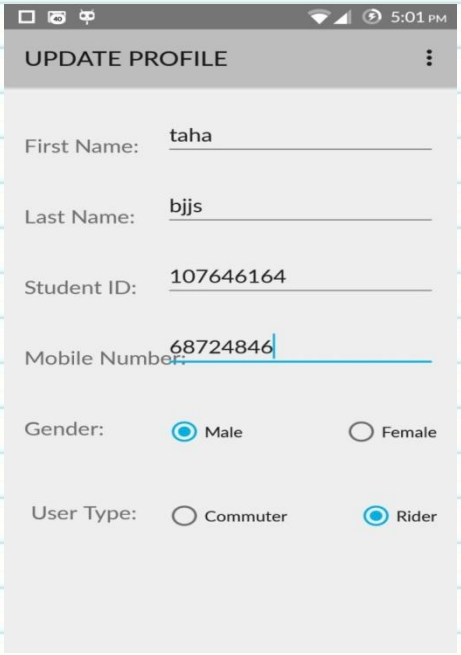
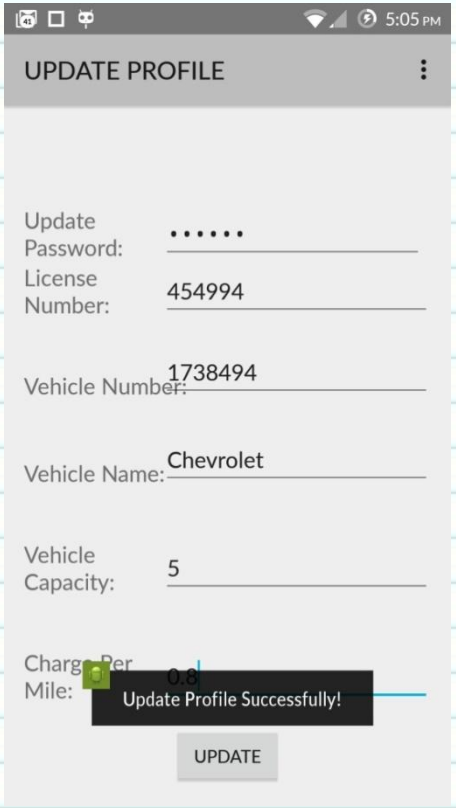


Figure 12:



Example Student Project (cont.)

Figure 13:

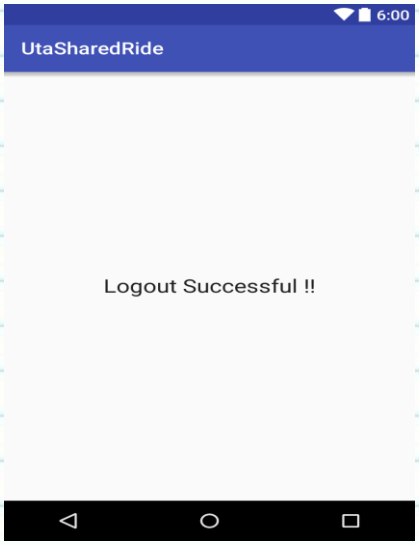
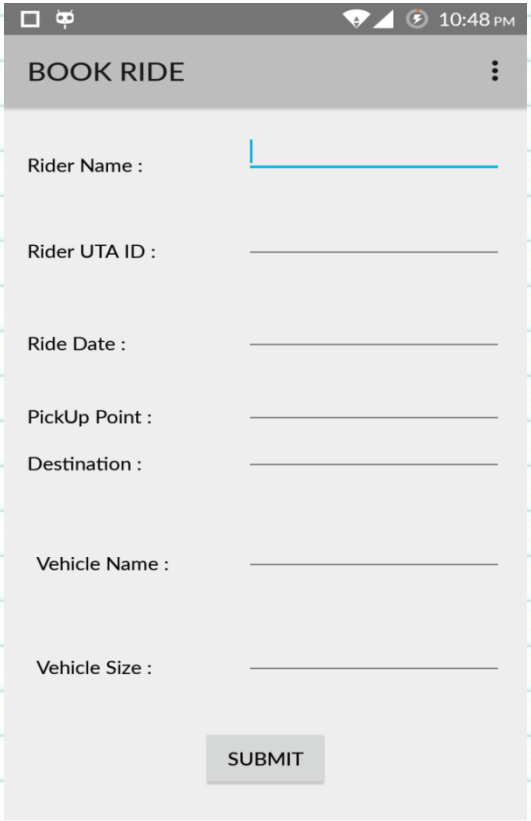


Figure 14:



Example Student Project (cont.)

Figure 15:

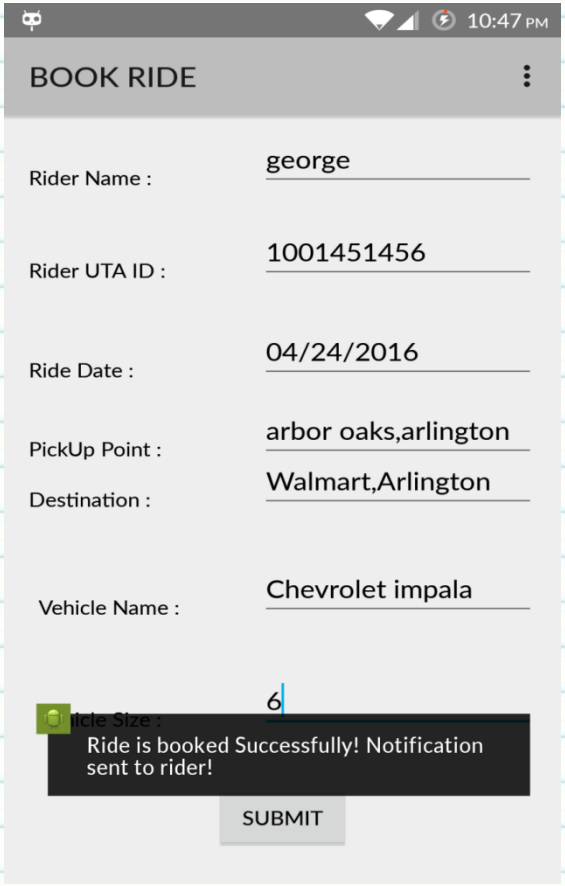
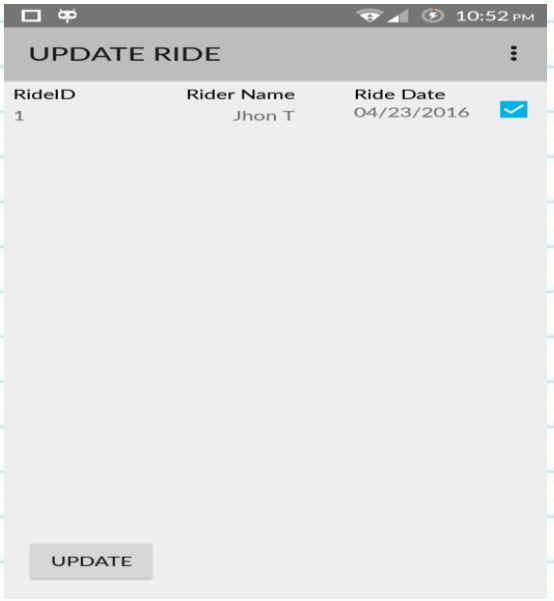


Figure 16:



Example Student Project (cont.)

Figure 17:

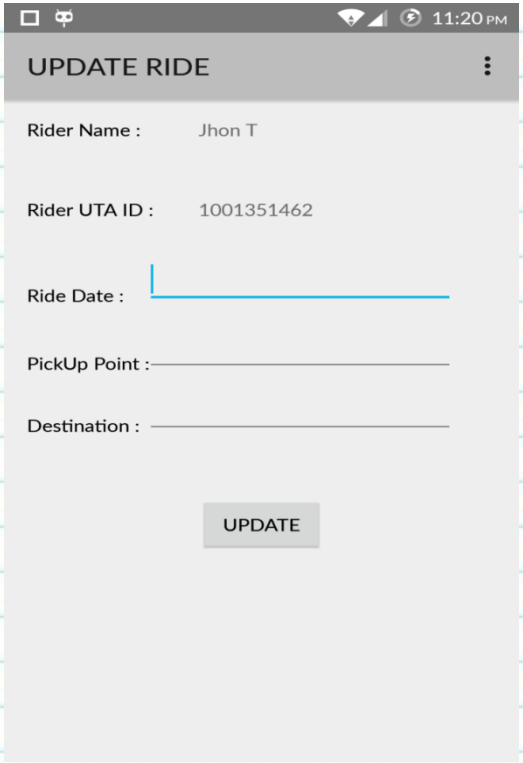
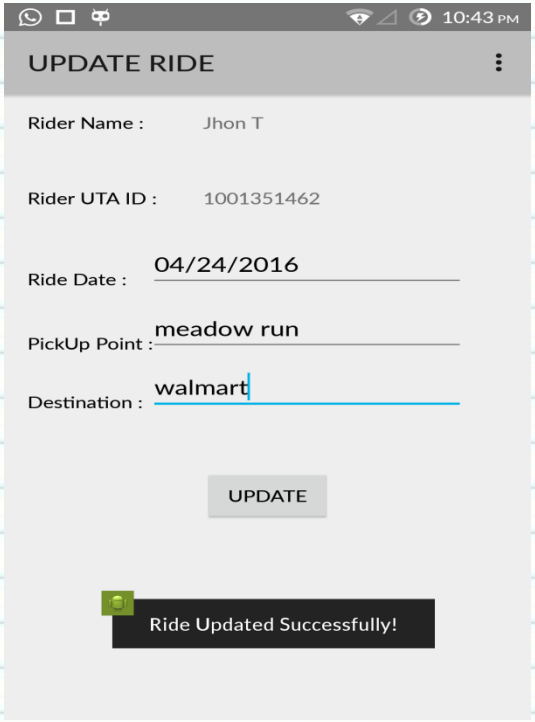


Figure 18:



Example Student Project (cont.)

Figure 19:

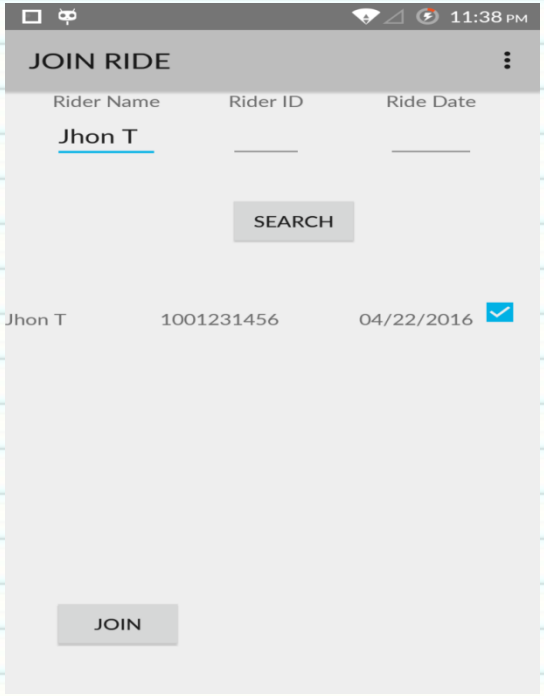


Figure 20: