CSE 324: Software Development sessional

Children Monitoring System
Technical documentation

Group members:

1. Rasheduzzaman (0805045)
2. Mohammad Minhazul Haq (0805051)

Subsystems & Actors

Subsystems

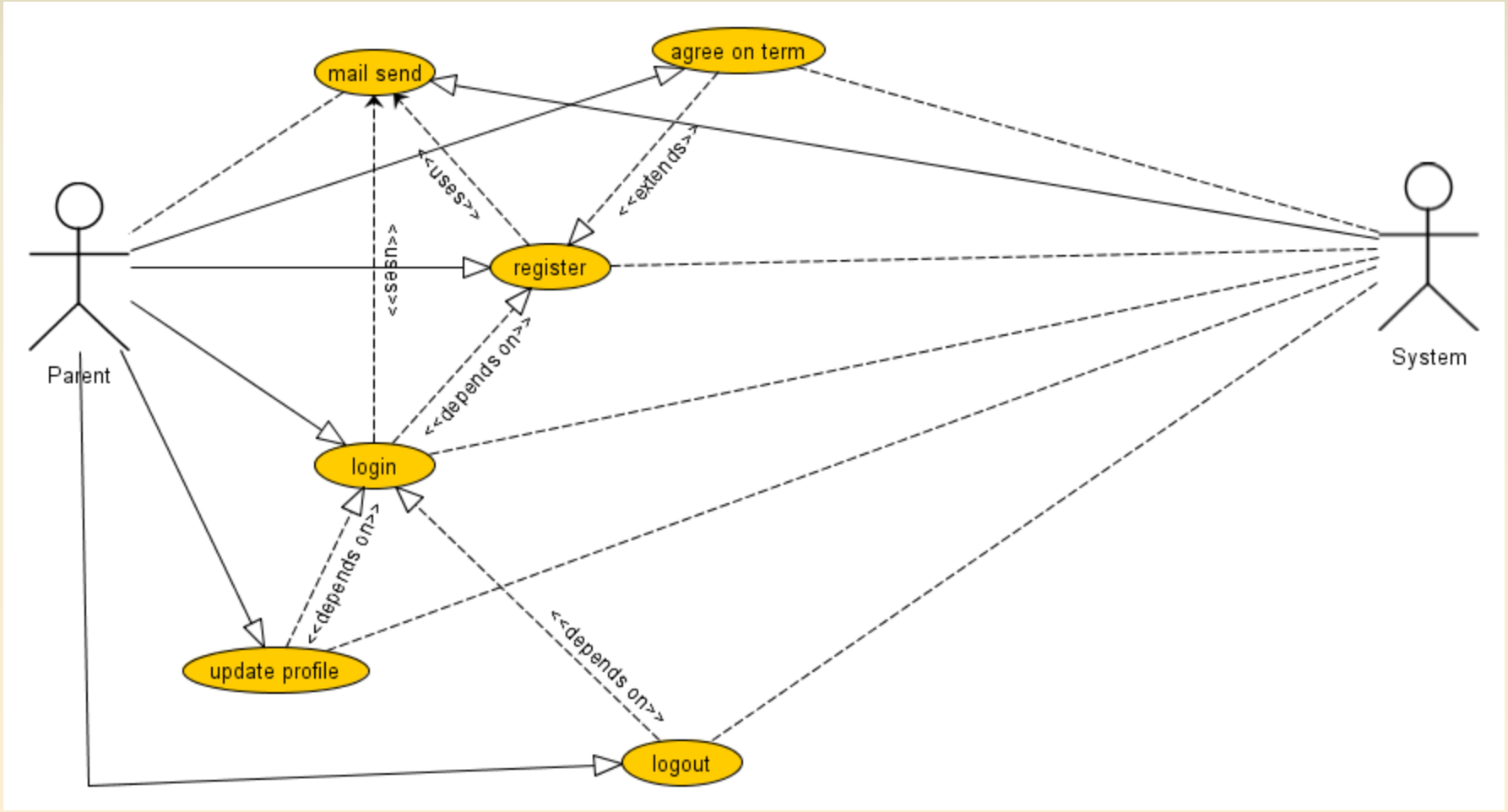
- Account subsystem
- Monitoring subsystem
- Application subsystem

Actors

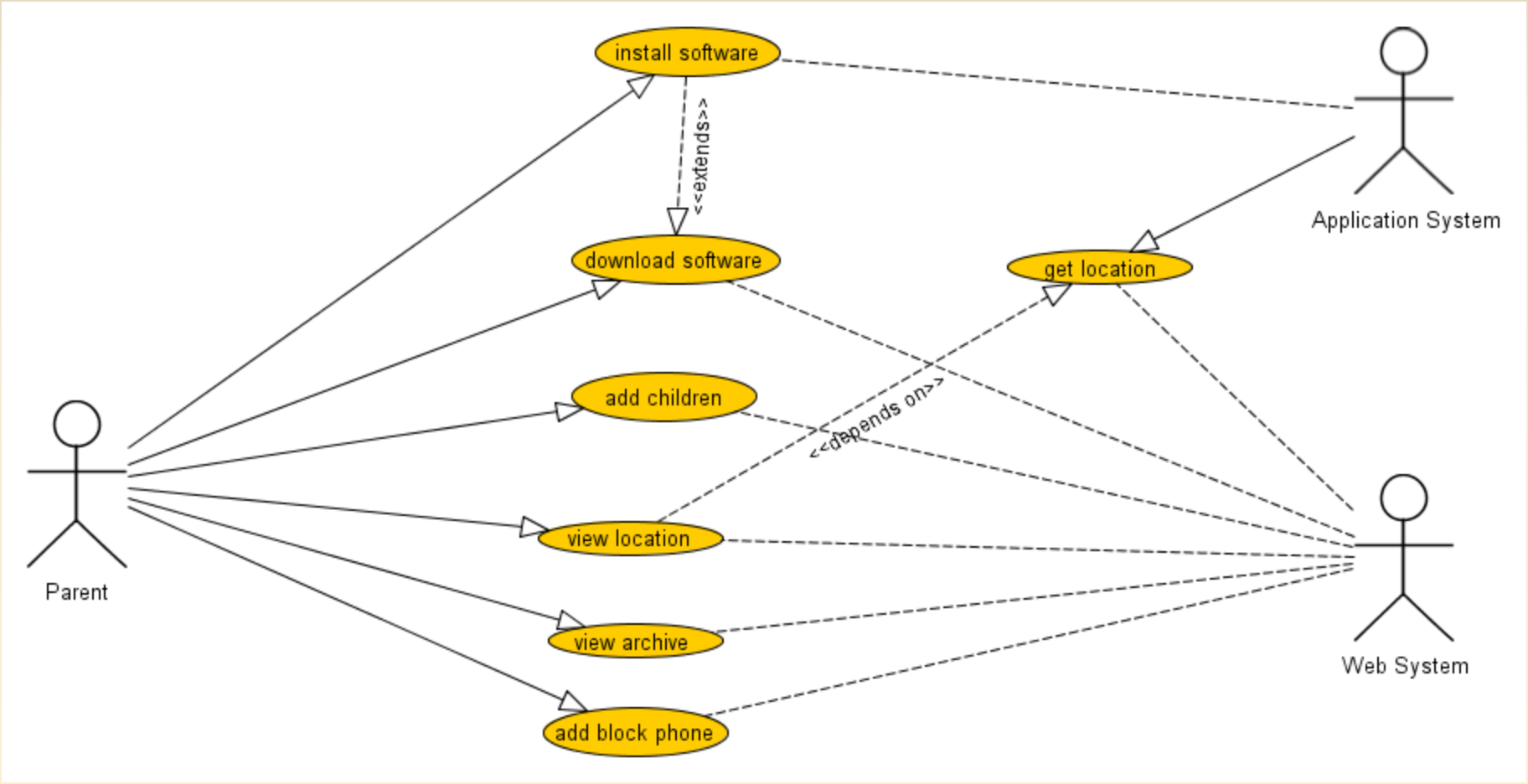
- Parent
- Web system
- Application system

Use-Case diagrams

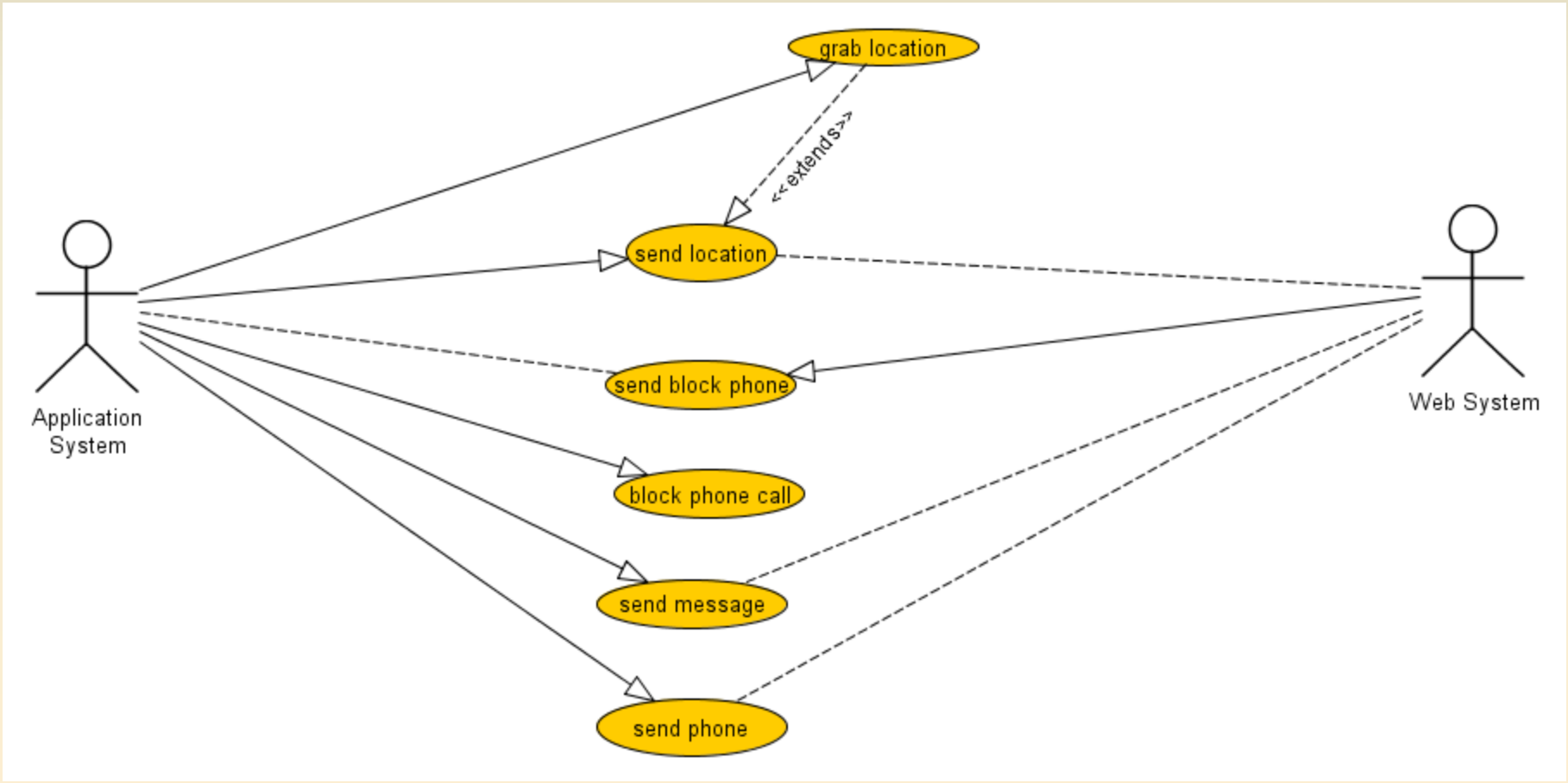
Account subsystem



Monitoring subsystem



Application subsystem



Use-Case narratives

Use-Case narrative: Account subsystem

Use Case Name:	login
Use Case Id:	UCIDAS 01
Priority:	High
Primary Business Actor:	Parent
Other Participating Actors:	Web System
Description:	This use case define the login mechanism of the website.

Use-Case narrative: Account subsystem

Precondition:	Must be a registered member	
Trigger:	Parent visit to the login page	
Typical Course Of events:	Actor Actions	System Actions
	Step1: provide the username and password.	Step2:verify username and password Step3: show the home page
Alternate Courses:	Alt-Step2:In case of failing to verify send login failed message with appropriate error message within it.	

Use-Case narrative: Account subsystem

Conclusion:	This use case concludes with the successful login into website.
Post condition:	Show the home page with home page data.
Implementation Constrains And Specification:	GUI for login.

Use-Case narrative: Account subsystem

Use Case Name:	register
Use Case Id:	UCIDAS 02
Priority:	High
Primary Business Actor:	Parent
Other Participating Actors:	Web System
Description:	This use case define the registration mechanism of the website.

Use-Case narrative: Account subsystem

Precondition:	Must be a parent	
Trigger:	Parent visit to the registration page	
Typical Course of events:	Actor Actions	System Actions
	Step1: provide the username , password,email and other data to the server Step5: click the link provided by email	Step2:verify the data Step3: insert the data into table Step4: sent an email with a link Step6: activate the account
Alternate Courses:	Alt-Step2:In case of failing to verify the entered data , send an appropriate error message Alt-Step4: In case of failing to send the email, send an appropriate error message and try to send again	

Use-Case narrative: Account subsystem

Conclusion:	This use case concludes with the successful user registration
Post condition:	Activate the user account
Implementation Constrains and Specification:	GUI for registration

Use-Case narrative: Monitoring subsystem

Use Case Name:	View location
Use Case Id:	UCIDAS 04
Priority:	High
Primary Business Actor:	Parent
Other Participating Actors:	System
Description:	This use case define how to see the location of his registered kid

Use-Case narrative: Monitoring subsystem

Precondition:	Must be logged in	
Trigger:	User opt to see the location in the map	
Typical Course Of events:	Actor Actions	System Actions
	Step1: press the “see location” button. Step3: provide the child name.	Step2: prompt to give the registered child name . Step4: get current location from database with provided child name Step5: provide the location in google map.
Alternate Courses:	Alt-Step4:In case of not getting the current location, then grab the most recent location data Alt-Step5:In case of problem in google map show the latitude and longitude value to the user	

Use-Case narrative: Monitoring subsystem

Conclusion:	This use case concludes with the successful location in map view operation.
Post condition:	User select another time's location.
Implementation Constrains and Specification:	GUI and google map to be provided to display current location.

Use-Case narrative: Monitoring subsystem

Use Case Name:	Download software
Use Case Id:	UCIDAS 05
Priority:	High
Primary Business Actor:	Parent
Other Participating Actors:	System
Description:	This use case define how to download the application from the server

Use-Case narrative: Monitoring subsystem

Precondition :	Must be a registered and verified member	
Trigger :	User opt to download the application from the server	
Typical Course Of events :	Actor Actions	System Actions
	Step1: press the “download” button. Step3: agree the license. Step5: provide the phone number and child name.	Step2: prompt to accept the license agreement. Step4: prompt to provide the phone number and the child name where the app will be installed. Step6: save the phone number and child name in database. Step7: start to download.
Alternate Courses :	Alt-Step4:In case of not agreeing the license agreement show the error message. Alt-Step6:In case of problem in downloading show the error message and prompt to download again.	

Use-Case narrative: Monitoring subsystem

Conclusion:	This use case concludes with the successful download of the application.
Post condition:	User install the software in the Phone.
Implementation Constrains and Specification:	GUI to be provided to display the download page and the software file should be uploaded in the server.

Use-Case narrative: Application subsystem

Use Case Name:	Send block phone
Use Case Id:	UCIDAS 010
Priority:	High
Primary Business Actor:	App System
Other Participating Actors:	Web System
Description:	This use case define the sending of blocked phone numbers to the child's phone

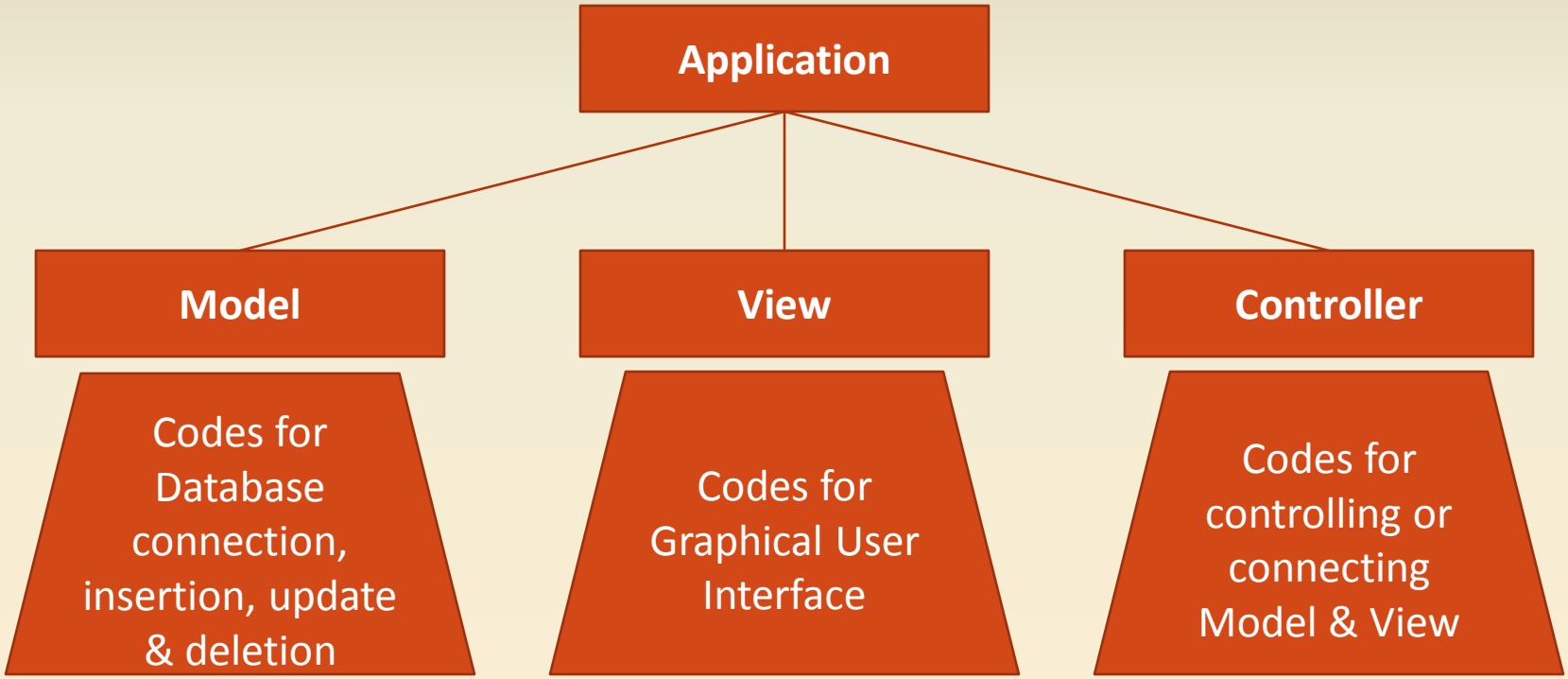
Use-Case narrative: Application subsystem

Precondition:	Must have the phone number in database	
Trigger:	App System send request for the phone number list	
Typical Course of events:	Actor Actions	System Actions
	Step1: send the request of the phone number list along with the phone number of itself. Step4: save the list in the local database. Step5: send an acknowledgement.	Step2: find out the list of that phone number from database. Step3: send the list to the application.
Alternate courses:	Alt-Step2:In case of not finding the list send an appropriate error message.	

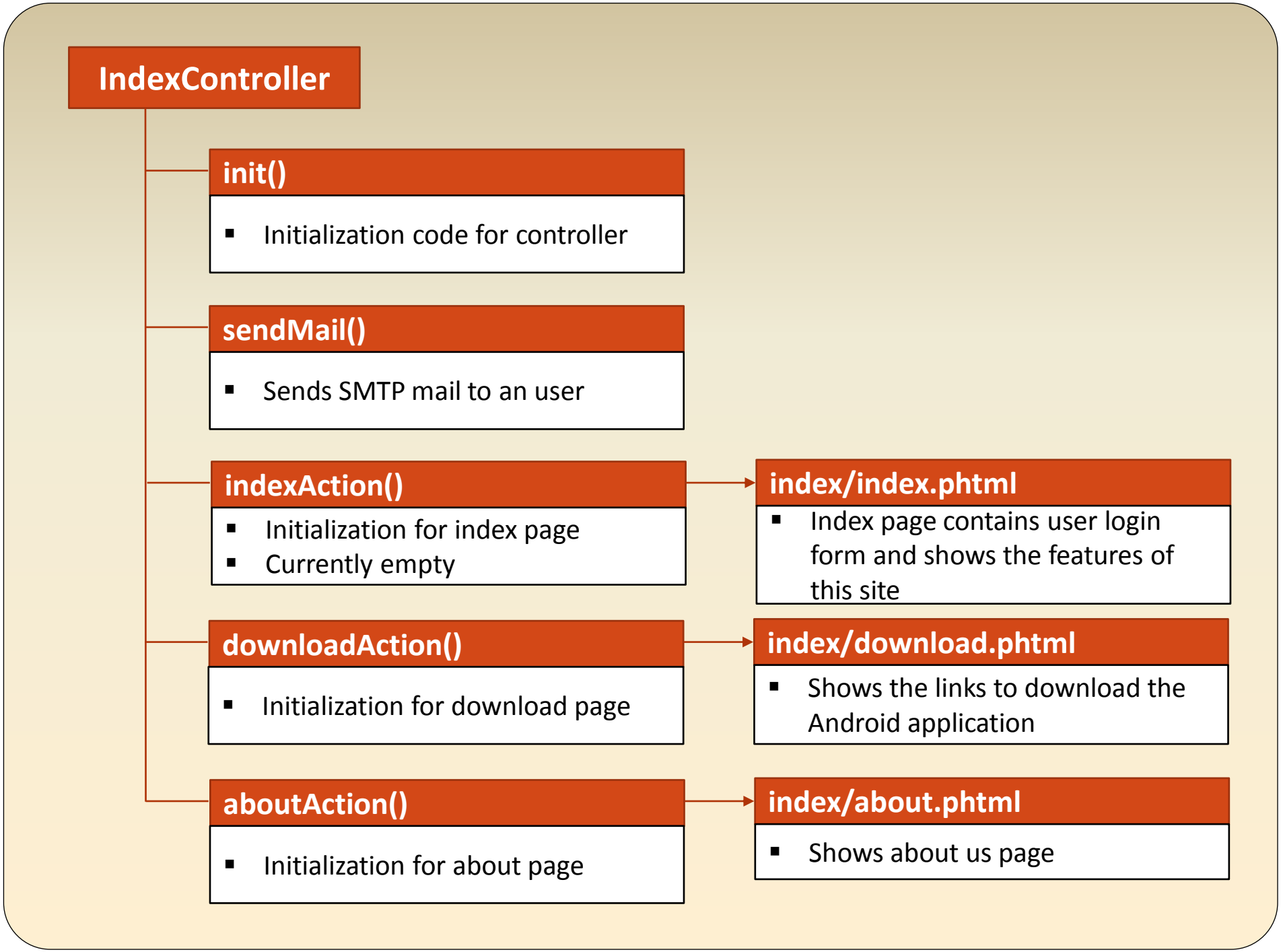
Use-Case narrative: Application subsystem

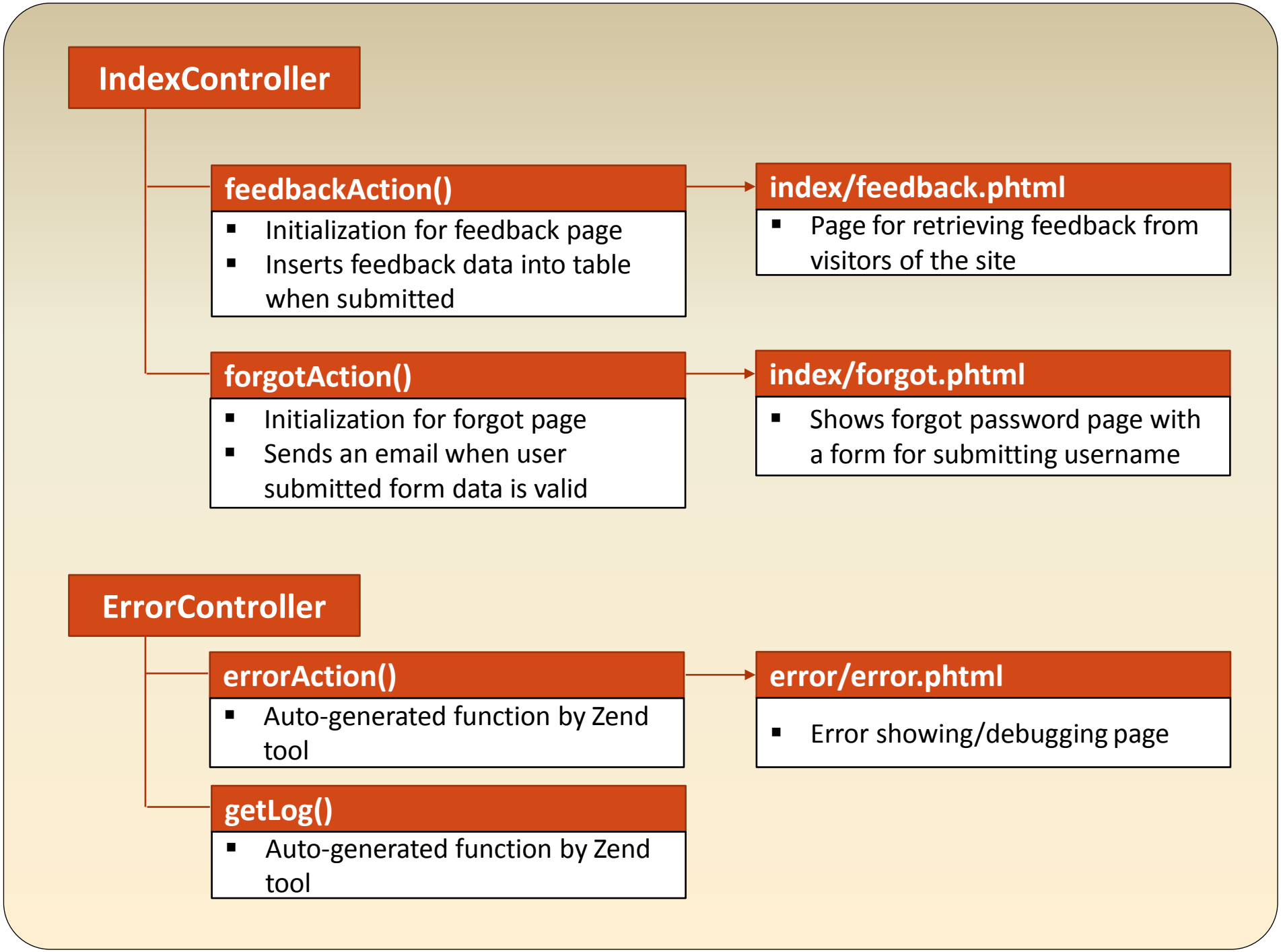
Conclusion:	This use case concludes with the successful insertion of blocked phone number list into database.
Post condition:	Block the phone call which number being in the list.
Implementation Constrains and Specification:	Need internet connection so that data can be passed to server and vice-varsa.

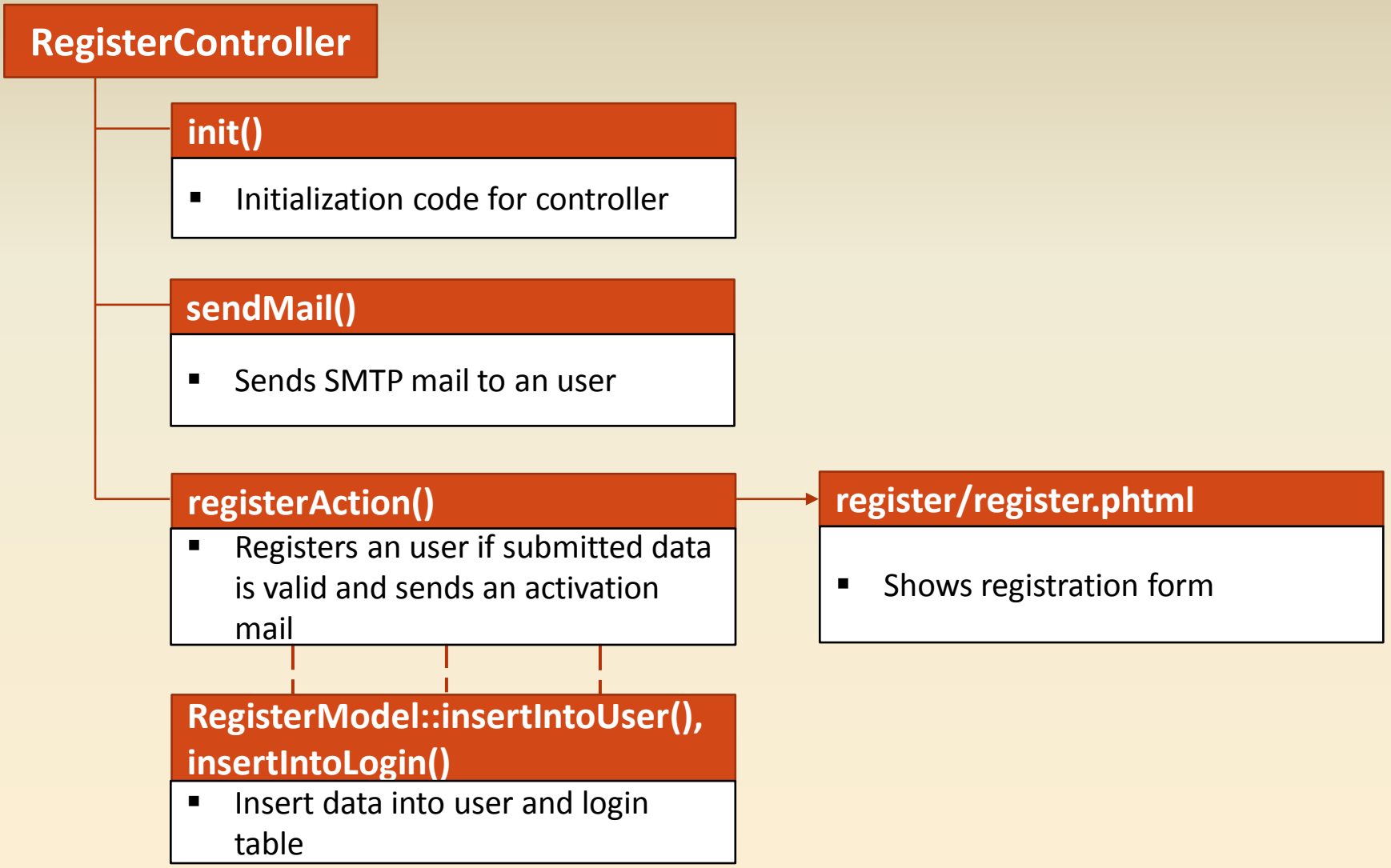
Model-View-Controller

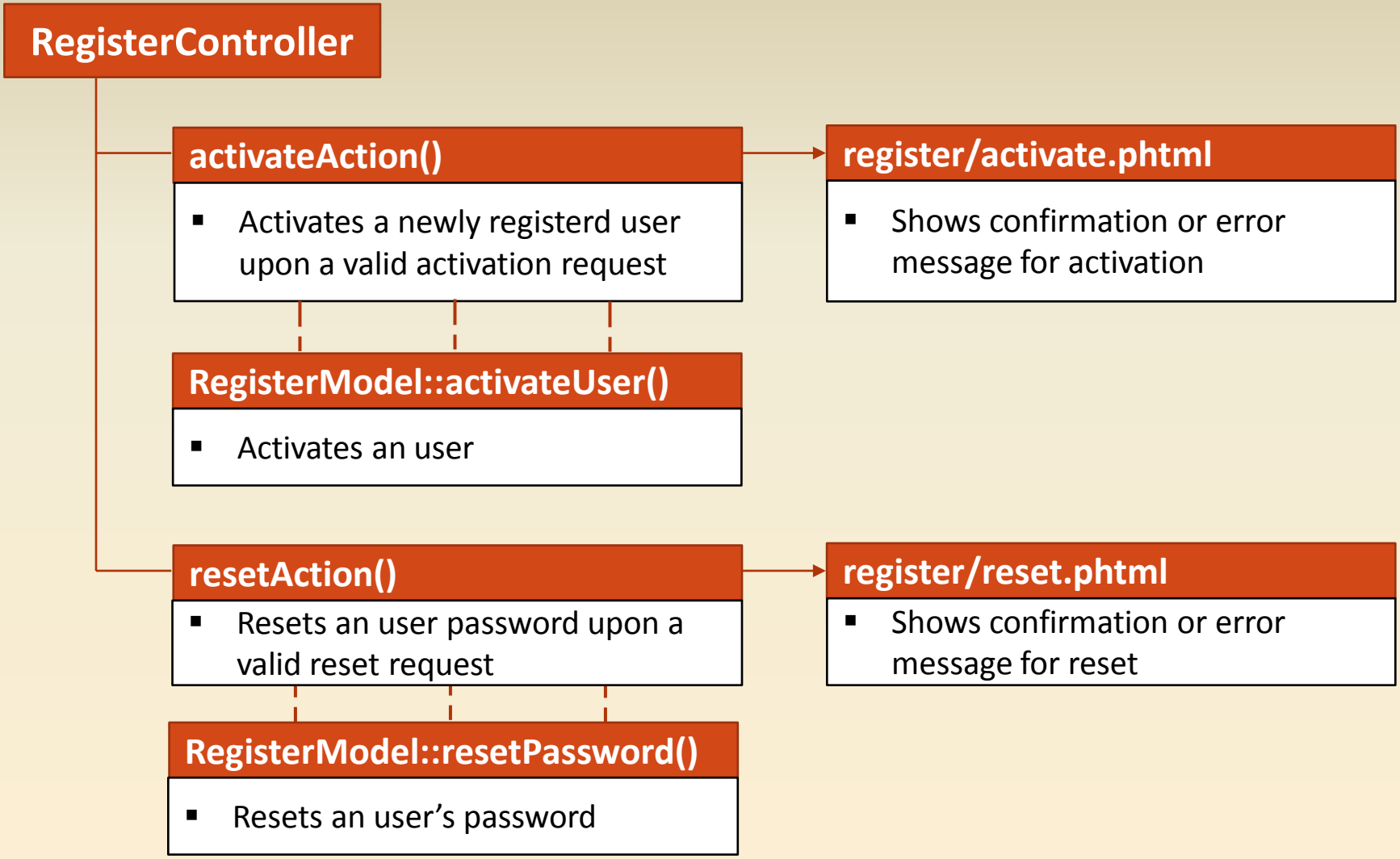

Application
Model
View
Controller
Codes for Database connection, insertion, update & deletion
Codes for Graphical User Interface
Codes for controlling or connecting Model & View

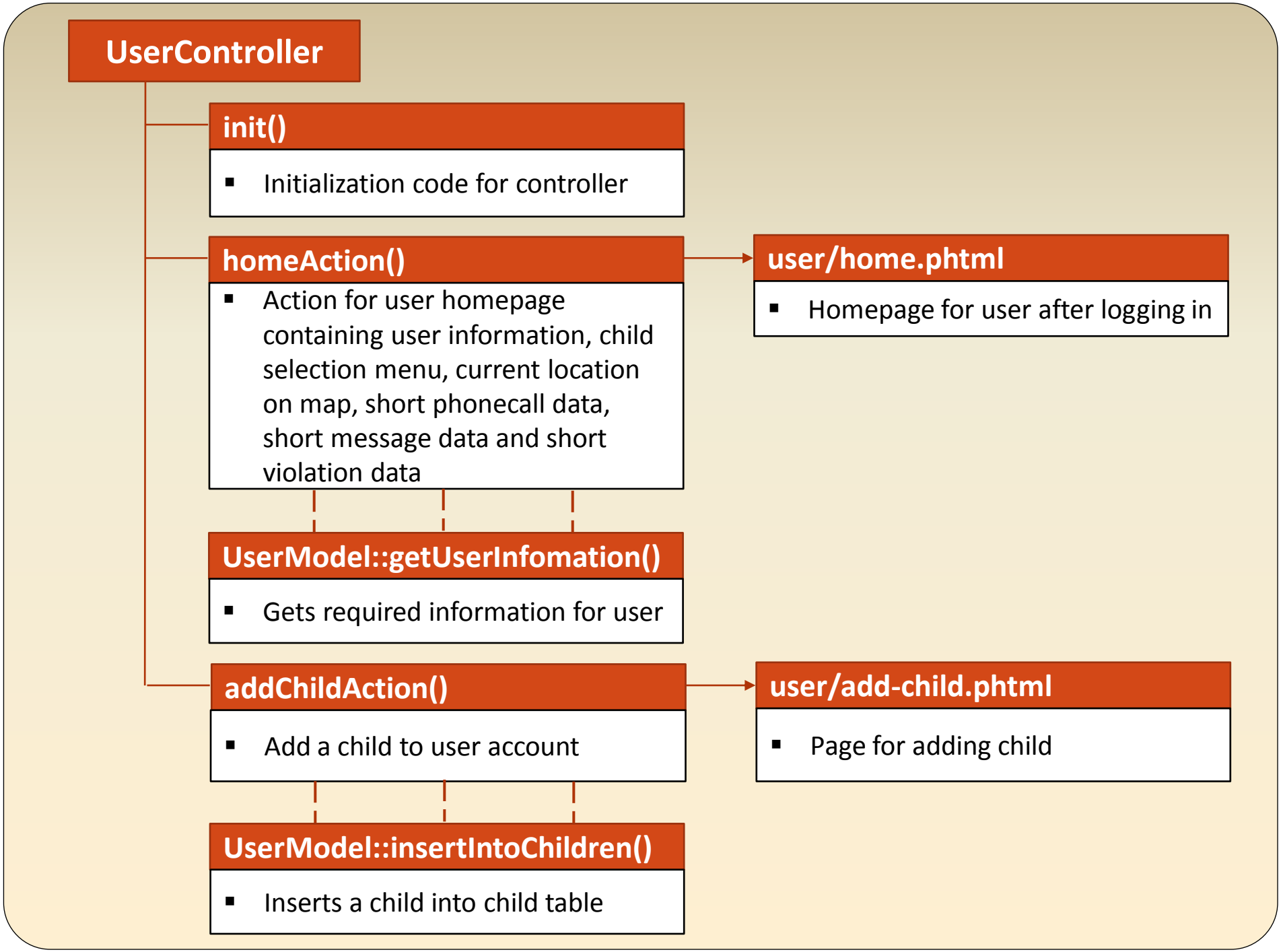
Class & functions for Server

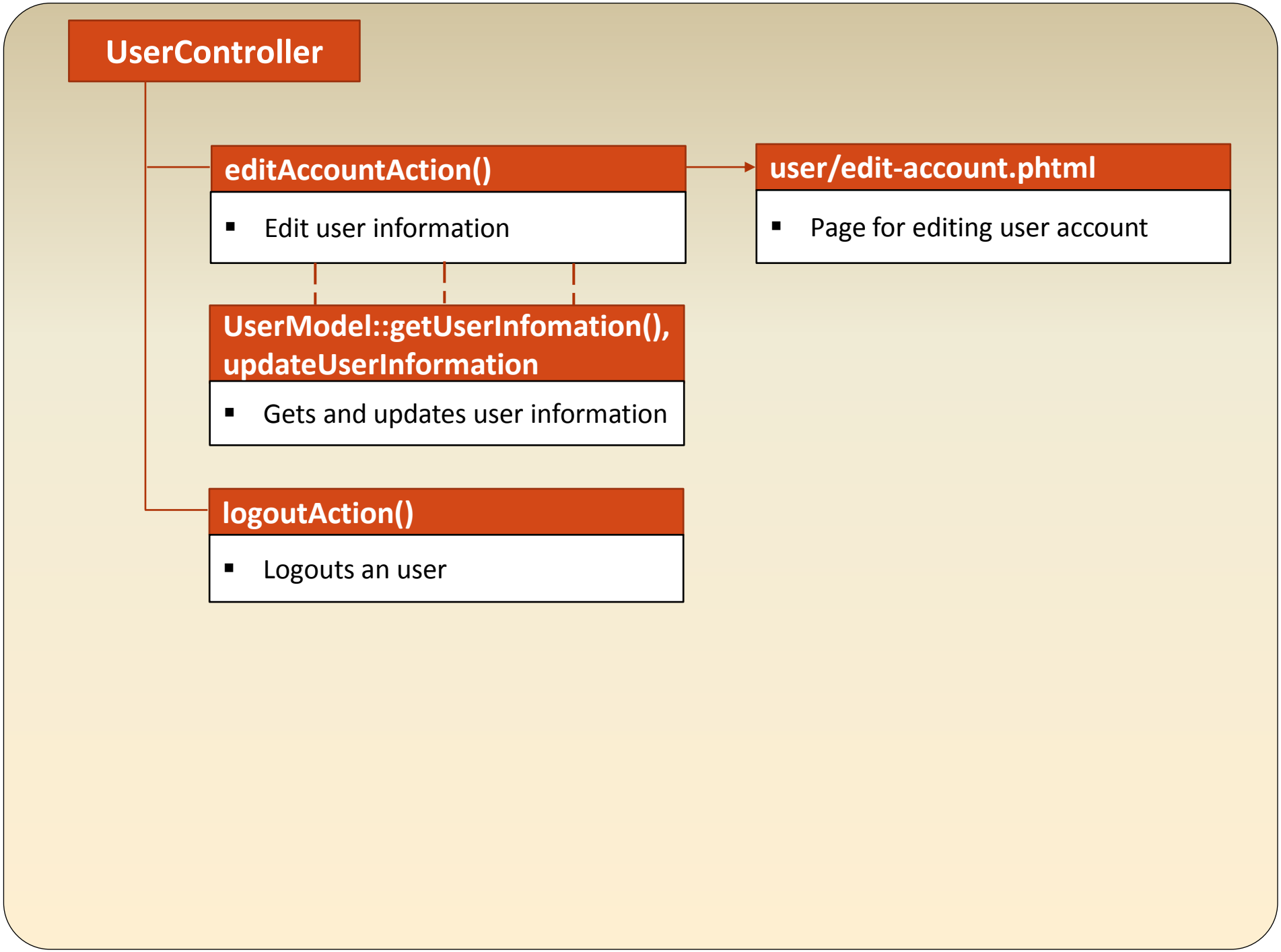

IndexController
init()
Initialization code for controller
sendMail()
Sends SMTP mail to an user
indexAction()
Initialization for index page
Currently empty
index/index.phtml
Index page contains user login form and shows the features of this site
downloadAction()
Initialization for download page
index/download.phtml
Shows the links to download the Android application
aboutAction()
Initialization for about page
index/about.phtml
Shows about us page

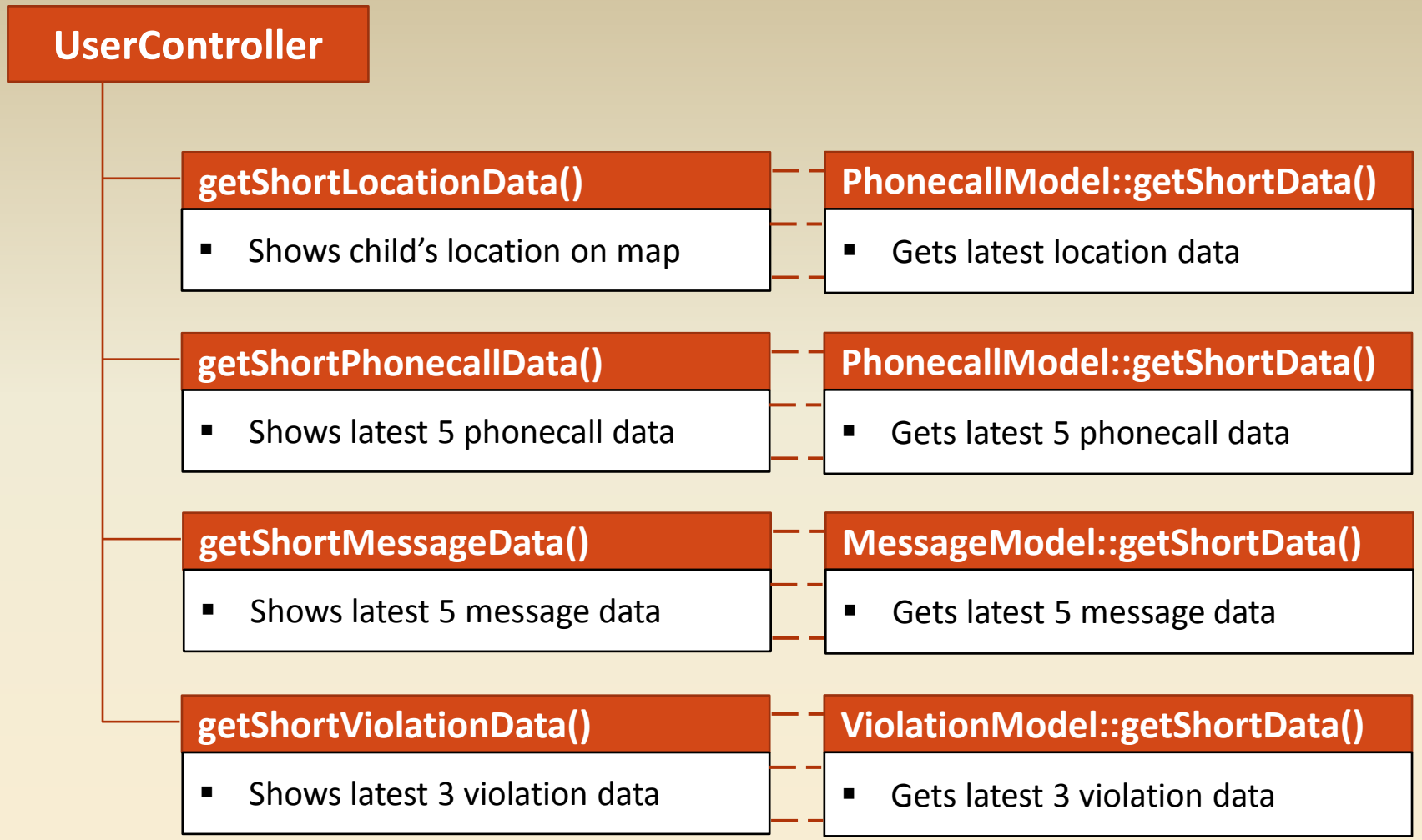

IndexController
feedbackAction()
Initialization for feedback page
Inserts feedback data into table when submitted
index/feedback.phtml
Page for retrieving feedback from visitors of the site
forgotAction()
Initialization for forgot page
Sends an email when user submitted form data is valid
index/forgot.phtml
Shows forgot password page with a form for submitting username
ErrorController
errorAction()
Auto-generated function by Zend tool
error/error.phtml
Error showing/debugging page
getLog()
Auto-generated function by Zend tool

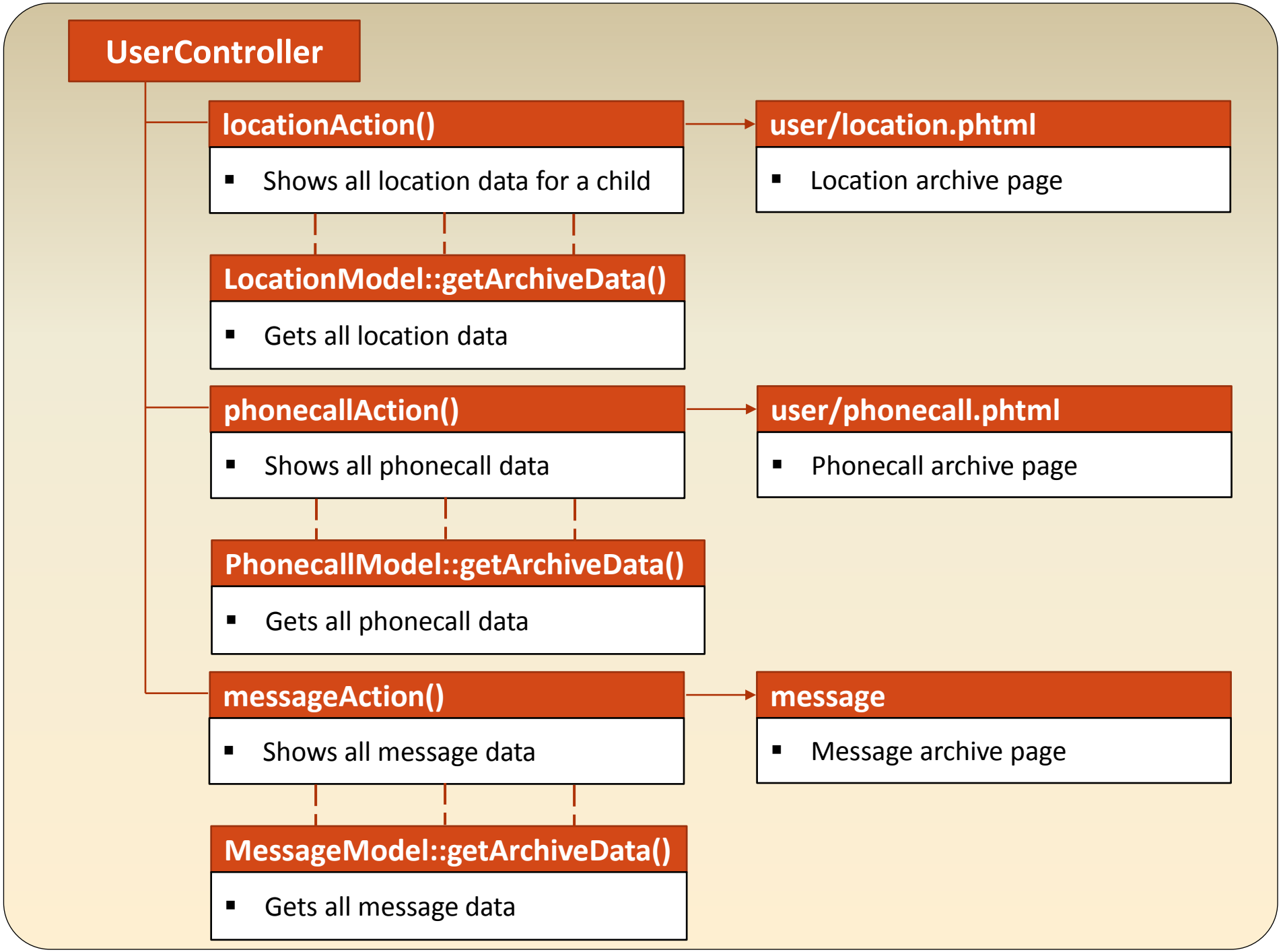

RegisterController
init()
Initialization code for controller
sendMail()
Sends SMTP mail to an user
registerAction()
Registers an user if submitted data is valid and sends an activation mail
register/register.phtml
Shows registration form
RegisterModel::insertIntoUser(), insertIntoLogin()
Insert data into user and login table

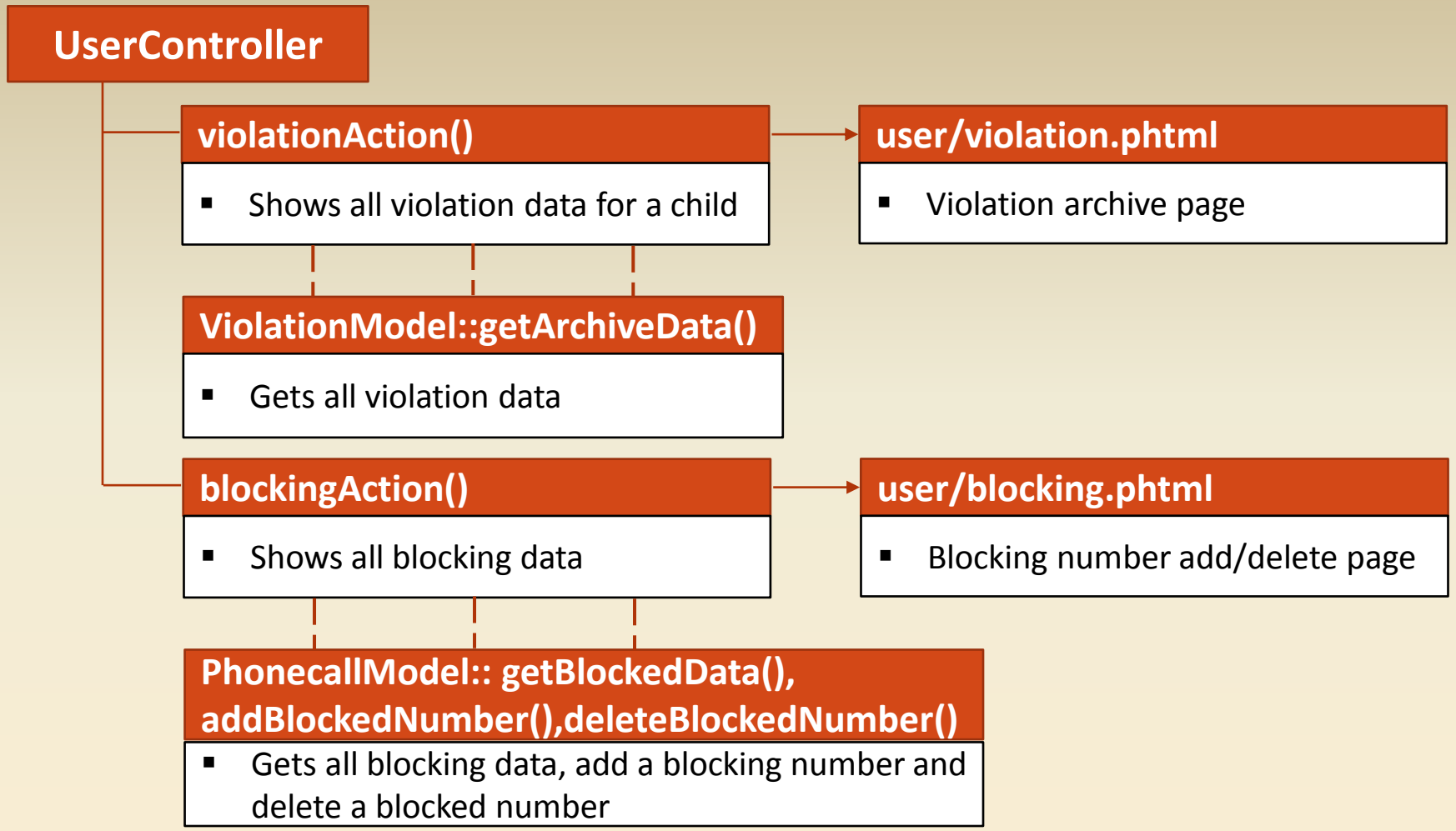

RegisterController
activateAction()
Activates a newly registerd user upon a valid activation request
register/activate.phtml
Shows confirmation or error message for activation
RegisterModel::activateUser()
Activates an user
resetAction()
Resets an user password upon a valid reset request
register/reset.phtml
Shows confirmation or error message for reset
RegisterModel::resetPassword()
Resets an user's password

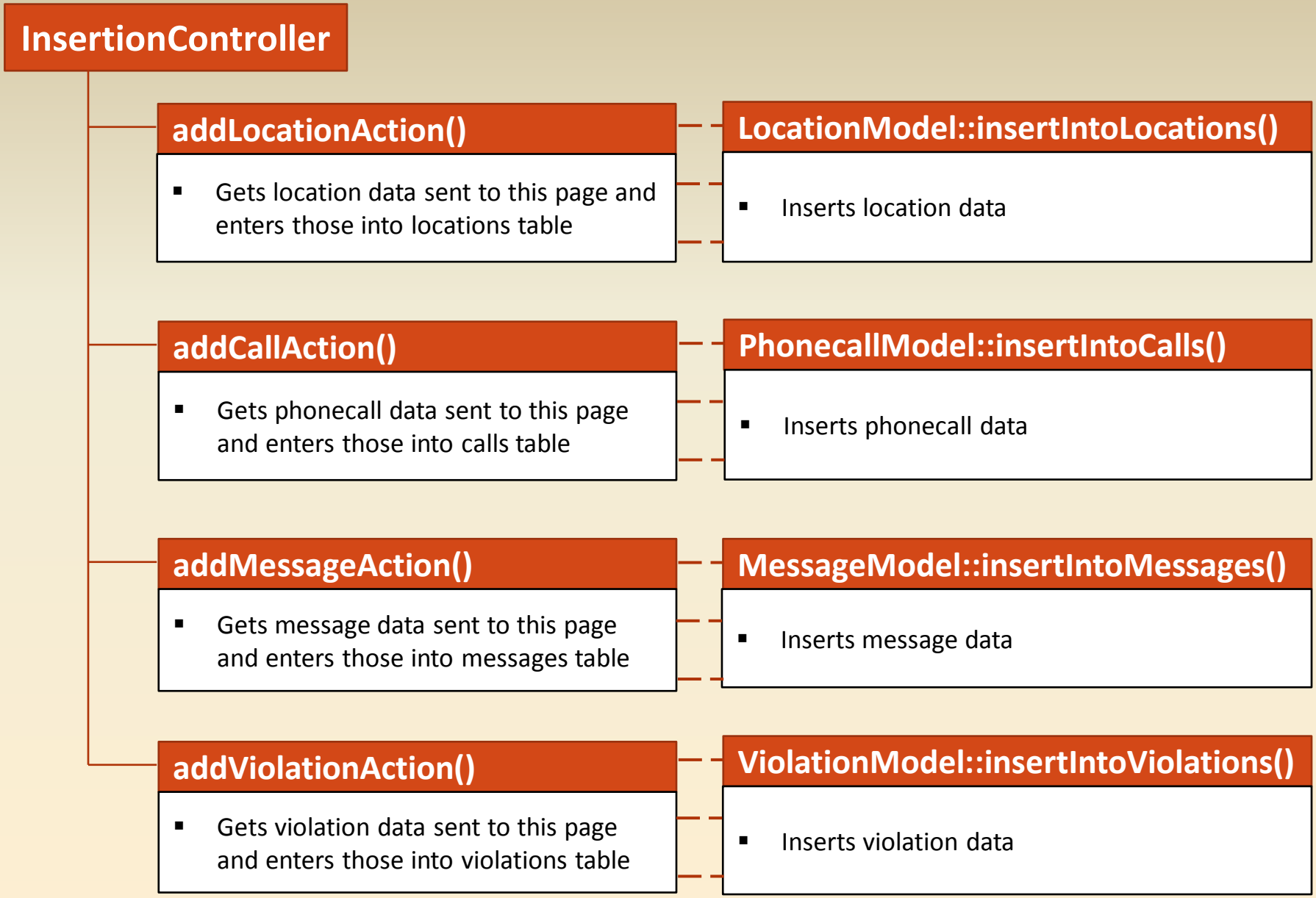

UserController
init()
Initialization code for controller
homeAction()
Action for user homepage containing user information, child selection menu, current location on map, short phonecall data, short message data and short violation data
user/home.phtml
Homepage for user after logging in
UserModel::getUserInfomation()
Gets required information for user
addChildAction()
Add a child to user account
user/add-child.phtml
Page for adding child
UserModel::insertIntoChildren()
Inserts a child into child table


UserController
editAccountAction()
Edit user information
user/edit-account.phtml
Page for editing user account
UserModel::getUserInfomation(), updateUserInformation
Gets and updates user information
logoutAction()
Logouts an user


UserController
getShortLocationData()
Shows child's location on map
PhonecallModel::getShortData()
Gets latest location data
getShortPhonecallData()
Shows latest 5 phonecall data
PhonecallModel::getShortData()
Gets latest 5 phonecall data
getShortMessageData()
Shows latest 5 message data
MessageModel::getShortData()
Gets latest 5 message data
getShortViolationData()
Shows latest 3 violation data
ViolationModel::getShortData()
Gets latest 3 violation data


UserController
locationAction()
Shows all location data for a child
user/location.phtml
Location archive page
LocationModel::getArchiveData()
Gets all location data
phonecallAction()
Shows all phonecall data
user/phonecall.phtml
Phonecall archive page
PhonecallModel::getArchiveData()
Gets all phonecall data
messageAction()
Shows all message data
message
Message archive page
MessageModel::getArchiveData()
Gets all message data


UserController
violationAction()
Shows all violation data for a child
user/violation.phtml
Violation archive page
ViolationModel::getArchiveData()
Gets all violation data
blockingAction()
Shows all blocking data
user/blocking.phtml
Blocking number add/delete page
PhonecallModel:: getBlockedData(), addBlockedNumber(),deleteBlockedNumber()
Gets all blocking data, add a blocking number and delete a blocked number


InsertionController
addLocationAction()
Gets location data sent to this page and enters those into locations table
LocationModel::insertIntoLocations()
Inserts location data
addCallAction()
Gets phonecall data sent to this page and enters those into calls table
PhonecallModel::insertIntoCalls()
Inserts phonecall data
addMessageAction()
Gets message data sent to this page and enters those into messages table
MessageModel::insertIntoMessages()
Inserts message data
addViolationAction()
Gets violation data sent to this page and enters those into violations table
ViolationModel::insertIntoViolations()
Inserts violation data

Class & functions for Android application's

Package: com.minhazrashed.childrenmonitoringsystem

MainActivity class

- **Action:** Starts the “MainService” and then disappear.
- **Extented class:** android.app.Activity
- **Field summary:** android.content.Intent
- **Methods inherited from android.app.activity:**
  setContentView, startService, stopService, stopSelf

MainActivity

void onCreate (Bundle savedInstanceState)

- Called when activity started

boolean startMainService (Intent intent)

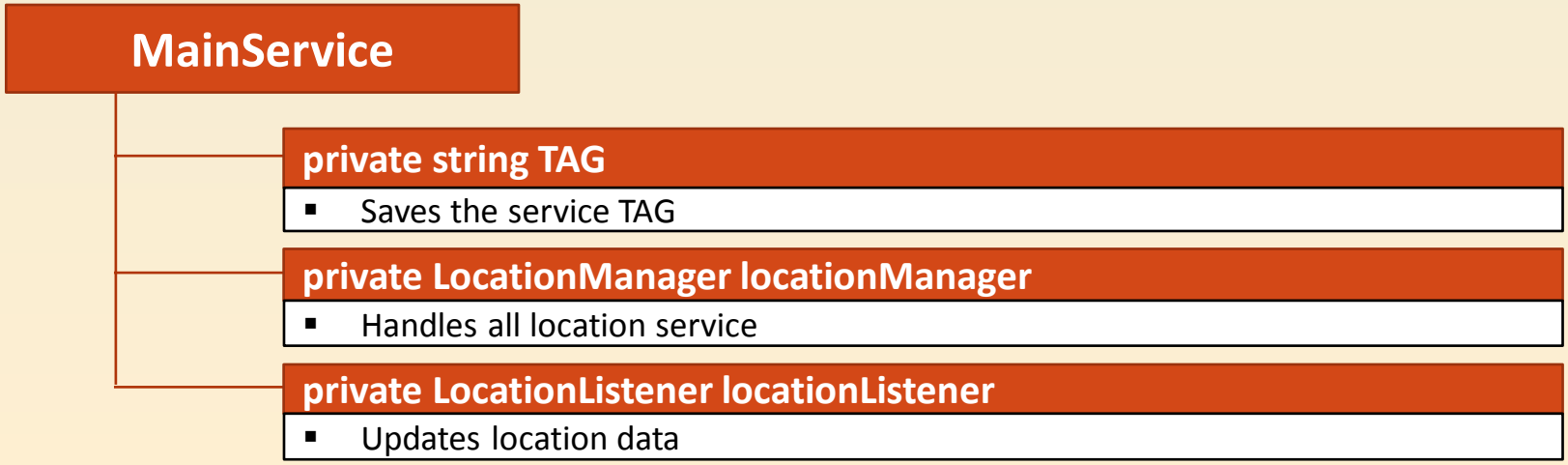
- Starts the service “Main Service”

void stopMainActivity (void)

- Stops MainActivity

MainService class

- **Action:** This service will run in the background in whole time and operate all operation in the application.
- **Extented class:** android.app.Service
- **Implemented class:** android.location.LocationListener
- **Methods inherited from android.app.activity:**
  onCreate, onStart, onDestroy, onBind
- **Methods inherited from android.app.activity:**
  onLocationChanged,onProviderDisabled, onProviderEnabled, onStatusChanged



MainService

private string phoneNumber

- Saves the phone number

private MainHttpConnection mainHttpConnection

- Instance of the class

private MainDatabaseAdapter mainDatabaseAdapter

- Instance of the class

void onCreate()

- Called when service created

void onStart (Intent, int)

- Called when service started

String getPhoneNumber()

- Gets the phone number and save it in “phoneNumber”

void connectHttp()

- Connect to http server

void connectDatabase()

- Connect with the SQLite local database

void onLocationChanged (Location)

- Grab the location and sends it to server by calling MainHttpConnection.sendLocation()

MainHttpConnection class

- **Action:** This is a utility class. It will serve the "MainService" class by providing some utility method to communicate with server over internet.

MainHttpConnection

private string TAG
- Saves the service TAG

private Httpclient client
- Creates connection with server

private HttpPost request
- Sends a post request to server

void MainHttpConnection()
- Default constructor

void sendLocation (double, double, String)
- Sends location and phone number to server

void getBlockedNumber (String)
- Gets blocked phone number from server according to a phone number

void sendSMS (String, String, String)
- Sends the SMS and phone number to the server

MainDatabaseAdapter class

- **Action:** This a adapter class which will connect with database with the help of MainDatabaseOpenHelper.

MainDatabaseOpenHelper class

- **Action:** This is the class which will talk with local database directly.

End of slides